UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de máster

Audio generado frente a audio original: Análisis y comparación mediante técnicas de recuperación de información musical

Alberto Prudencio de Dueñas

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Proyecto para la obtención del título de
Máster en Investigación e Innovación en
Inteligencia Computacional y Sistemas
Interactivos
por la Universidad Autónoma de Madrid

Tutor del trabajo fin de máster:
Fernando Díez Rubio

Audio generado frente a audio original: Análisis y comparación mediante técnicas de recuperación de información musical

Alberto Prudencio de Dueñas

Alberto Prudencio de Dueñas
Audio generado frente a audio original: Análisis y comparación mediante técnicas de recuperación de información musical

Alberto Prudencio de Dueñas
C\ del Balandro Nº 5

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos.

In the midst of chaos, there is also opportunity.

Sun-Tzu

Agradecimientos

A mis padres, habéis sido una fuente de apoyo incondicional en la que siempre he confiado. Gracias por estar siempre ahí y darme consejo cuando lo he necesitado.

A mis hermanos, que a pesar de pasar poco tiempo con ellos, para mí son un ejemplo a seguir. Me habéis cuidado toda la vida como si fuera vuestro hijo y no podría estaros más agradecido de todo lo que me habéis enseñado. Soy quién soy gracias a vosotros, y estoy muy orgulloso de ello.

A mi novia y amigos, siempre atentos y dispuestos a ayudar. Muchas gracias por cuidarme incluso en mis peores momentos, sin vuestro apoyo este trabajo no hubiera sido posible.

Por último, mencionar la excelente relación mantenida con el tutor. Desde el primer momento nos hemos compenetrado y ayudado mutuamente. Gracias por depositar tu confianza en mí, espero que hayas disfrutado tanto como yo de este trabajo.

Resumen

A lo largo de este trabajo se plantea el estudio sobre las diferencias existentes entre audio original, creado por un humano, y audio artificial, generado por un ordenador. Para llevar a cabo esta comparación se utilizan herramientas de composición artificial y de recuperación de información musical.

El problema de la creación de audio está tomando auge en los últimos años. Es un problema que cada vez se está estudiando en mayor profundidad, empleando para ellos nuevas metodologías y tecnologías que exploraremos. Para aplicar estas, revisaremos el estado del arte y estableceremos una serie de herramientas a utilizar.

De cara a estudiar estas diferencias, realizamos un experimento de generación musical, del que explicaremos con detalle los pasos llevados a cabo. A través de las generaciones obtenidas, extraeremos una serie de características que nos permitirán analizar el problema planteado. Para reforzar el análisis sobre las generaciones obtenidas se plantea una encuesta musical en la que los usuarios deben diferenciar si las melodías que están escuchando son compuestas por un humano o por un ordenador.

Finalizado el análisis, discutiremos sobre el problema planteado a través de los aspectos más relevantes del trabajo y la información obtenida del análisis. Por último, estableceremos las conclusiones obtenidas haciendo un repaso de los objetivos y puntos más importantes tratados en la discusión. Junto a las conclusiones trataremos aquellos aspectos a mejorar de cara al trabajo futuro.

Palabras clave

Composición artificial, Recuperación de información musical, Redes neuronales, Música, Magenta, MuseGAN, Musicnn, Reconocimiento de género, Etiquetado Musical

Abstract

Throughout this work, the study on the differences between original audio, created by a human, and artificial audio, generated by a computer, is proposed. We use artificial composition and musical information recovery tools to compare human against computer performance.

The problem of audio creation has been growing in recent years. It is a problem that is constantly being studied in greater depth, using new methodologies and technologies that we will explore. To apply these, we will review the state of the art and establish a series of tools to be used.

In order to study these differences, we will carry out a music generation experiment, from which we will explain in detail the steps taken. Through the generations obtained, we will extract a series of characteristics that will allow us to analyze the problem presented. To reinforce the analysis of the generations obtained, we propose a musical survey in which users must differentiate whether a human or a computer composes the tunes they are listening to.

Once the analysis is complete, we will discuss the problem posed through the most relevant aspects of the work and the information obtained from the analysis. Finally, we will establish the conclusions obtained by reviewing the objectives and most important points discussed. Together with the conclusions, we will deal with those aspects to be improved for future work.

Keywords

Artificial Composition, Musical Information Retrieval, Neural Networks, Music, Magenta, MuseGAN, Musicnn, Genre Recognition, Music Tagging

Índice

1 Introducción **1**

1.1 Motivación . . . 1

1.2 Objetivos . . . 2

1.3 Organización de la memoria . . . 3

2 Estado del arte **5**

2.1 Composición artificial . . . 5

2.1.1 Modelos algorítmicos . . . 5

2.2 Recuperación de información musical . . . 7

2.2.1 Transformada de Fourier de tiempo reducido (STFT) . . . 8

2.2.2 Espectrograma de Mel (MFCC) . . . 9

2.2.3 Transformación Q constante (CQT) . . . 10

2.2.4 Cromagrama . . . 11

2.3 Redes neuronales artificiales . . . 12

2.3.1 Aprendizaje Profundo (Deep Learning) . . . 12

2.3.2 Redes neuronales recurrentes (RNN) . . . 12

2.3.3 Redes neuronales convolucionales (CNN) . . . 15

2.3.4 Redes adversarias generativas (GAN) . . . 16

3 Tecnologías utilizadas **19**

3.1 Magenta . . . 19

3.1.1 Melody RNN . . . 20

3.1.2 Polyphony RNN . . . 21

3.1.3 Performance RNN . . . 22

3.2 MuseGAN . . . 22

3.3 Musicnn . . . 25

3.4 LSTM GClassifier . . . 25

4 Fase experimental **27**

4.1 Fase 1: Creación del conjunto de datos personalizado . . . 27

4.2 Fase 2: Entrenamiento y generación de composiciones . . . 33

4.2.1 Entrenamiento Magenta . . . 34

4.2.2 Generaciones Magenta . . . 36

4.2.3 Entrenamiento MuseGAN . . . 39

4.2.4 Generaciones MuseGAN 42
4.3 Fase 3: Extracción de características y obtención de resultados 43
4.3.1 Etiquetado musical mediante Musicnn 48
4.3.2 Reconocimiento de género mediante LSTM GClassifier 49
4.3.3 Encuesta musical 50
5 Análisis de los resultados **57**
5.1 Análisis etiquetado musical 58
5.2 Análisis reconocimiento de géneros 63
5.3 Análisis encuesta musical 66
6 Discusión **73**
7 Conclusiones y trabajo futuro **77**
7.1 Trabajo futuro 78
Bibliografía **85**
Apéndices **87**
A Script python para extracción de géneros **89**
B Gráficas MIR completas **91**
C Opiniones y discusiones sobre encuesta musical **93**

Listas

Lista de algoritmos

Lista de códigos

4.1 Código para evitar desbordes de VRAM en la GPU. 33
4.2 Fragmento de código para recorrer todos los ficheros de un directorio 45
4.3 Configuración multihilo en Python 45
4.4 Conversión MIDI a MP3 46
4.5 Conversión pianoroll a MIDI 47
4.6 Etiquetado musical mediante Musicnn 48

A.1 Script recuperación/separación géneros musicales por artista/género (1). 89
A.2 Script recuperación/separación géneros musicales por artista/género (2). 90

Lista de cuadros

3.1 Representación de un acorde do mayor con duración de negra en Polyphony RNN . . . 21

4.1 Géneros musicales recuperados por Artista mediante Spotify API y almacenados en fichero JSON. 29
4.2 Comando para transformar MIDIs a formato NoteSequences. 30
4.3 Comando para convertir los datos al modelo Melody RNN. 31
4.4 Comando para convertir los datos al modelo Polyphony RNN. 31
4.5 Comando para convertir los datos al modelo Performance RNN. 31
4.6 Comando para convertir los datos al formato pianoroll multipista. 32
4.7 Comando para entrenar las configuraciones lookback y attention (Melody RNN). 35
4.8 Comando para entrenar el modelo Polyphony RNN. 36
4.9 Comando para entrenar el modelo Performance RNN en su configuración *multiconditioned_performance_with_dynamics*. 36
4.10 Comando para generar melodías en las configuraciones lookback y attention (Melody RNN). 37
4.11 Comando para generar melodías en el modelo Polyphony RNN. 37

4.12 Comando para generar melodías en el modelo Performance RNN en su configuración *multiconditioned_performance_with_dynamics*. 38
4.13 Comando para crear un *experimento* en MuseGAN. 40
4.14 Comando para crear un experimento en MuseGAN. 41
4.15 Comandos para generar composiciones en el modelo MuseGAN. 42

5.1 Formato de almacenado de etiquetas musicales. 58

Lista de ecuaciones

2.1 Transformada de Fourier en tiempo reducido 8
2.2 Transformación de frecuencias a escala Mel 9
2.3 Cálculo de frecuencias centrales para CQT 10
2.4 Ancho espectral en filtros CQT 10
2.5 Escala cromagrama 11

5.1 Error cuadrático medio 57

Lista de figuras

2.1 Transformada de Fourier en tiempo reducido: espectrograma 8
2.2 Espectrograma de Mel 10
2.3 Espectrograma CQT 11
2.4 Cromagrama 11
2.5 Ejemplo de red neuronal 12
2.6 Neurona normal frente a neurona perteneciente a RNN 13
2.7 Neurona RNN a lo largo del tiempo 13
2.8 Estructura neurona LSTM 14
2.9 Estructura de una red neuronal convolucional 15
2.10 Proceso de convolución 16
2.11 Estructura de una red adversaria generativa 17

3.1 Ejemplo de melodía monofónica 20
3.2 Ejemplo de melodía polifónica 21
3.3 Representación MIDI en formato pianoroll 23
3.4 Melodía escrita mediante una herramienta piano roll en Ableton Live 23

4.1 Estructura general del experimento 27
4.2 Fase 1 del experimento en detalle 33

4.3 Fase 2 del experimento en detalle 43
4.4 Primera sección de la encuesta 51
4.5 Segunda sección de la encuesta 52
4.6 Tercera sección de la encuesta 54
4.7 Fase 3 del experimento en detalle 55

5.1 Comparación global en etiquetado musical MTT 59
5.2 Diferencias cuadráticas en etiquetado musical MTT 60
5.3 Comparación global en etiquetado musical MSD 61
5.4 Diferencias cuadráticas en etiquetado musical MSD 62
5.5 Comparación global en reconocimiento de género 64
5.6 Diferencias cuadráticas en en reconocimiento de género 65
5.7 Resultados medios grupo 1 en encuesta musical 66
5.8 Resultados medios grupo 2 en encuesta musical 67
5.9 Resultados medios grupo 3 en encuesta musical 68
5.10 Resultados medios grupo 4 en encuesta musical 69
5.11 Puntuaciones medias globales en encuesta musical 69
5.12 Comparativa puntuación/edad en melodía 3 70
5.13 Comparativa puntuación/edad en melodía 7 71
5.14 Géneros musicales reconocidos en la encuesta musical 71

B.1 Comparación global completa en etiquetado musical MTT 91
B.2 Comparación global completa en etiquetado musical MSD 92

Lista de tablas

4.1 Tabla de hiperparámetros por defecto en Magenta 35
4.2 Métricas de entrenamiento en Magenta 37
4.3 Valores métrica 'loss' en MuseGAN tras entrenamiento 42

5.1 Error cuadrático medio en etiquetado musical MTT 59
5.2 Error cuadrático medio en etiquetado musical MSD 61
5.3 Error cuadrático medio en en reconocimiento de género 63
5.4 Puntuaciones medias en encuesta musical 70

Lista de cuadros

1

Introducción

En este capítulo se va a introducir el trabajo realizado a través de la motivación detrás de este, así como los objetivos planteados al comienzo de su desarrollo y una breve explicación de la estructuración de esta memoria.

1.1. Motivación

La historia nos ha enseñado que desde el principio de los tiempos, el ser humano ha vivido en armonia con la música. Los primeros indicios que demuestran que la música ha estado entre nosotros datan de la era prehistórica. No se sabe a ciencia cierta la edad en la que ésta apareció, sin embargo su aparición se asocia a la de las pinturas rupestres, y por lo tanto al primer acercamiento del ser humano al arte. Las pinturas rupestres más antiguas datan del año 60000 AC por lo que se cree que alrededor de esta época los humanos comenzaron a experimentar con sus voces y con acompañamientos tales como palmadas o golpes con piedras o palos [1].

En su evolución a lo largo de la historia la música ha sido interpretada de muchas formas distintas, estando completamente influenciada por la época o contexto en el que se desarrollaba. En la actualidad esto no ha cambiado y la música sigue en constante desarrollo, existiendo todo tipo de estilos musicales, sonidos o instrumentos a partir de los cuales componer nuevas piezas musicales. Este desarrollo se debe en gran parte a la tecnología, ya que gracias a ella disponemos de herramientas que facilitan nuestra relación con la música. Podemos disfrutarla a partir de distintas plataformas de streaming, compartirla a través de las redes sociales existentes, producirla a través de software enfocado a la composición musical, etc.

Del mismo modo, existen en la actualidad campos relacionados con la música y la tecnología que están enfocados en tareas tales como la composición musical o la recuperación de información.

La composición artificial es un campo muy extenso que ha existido desde los comienzos de la informática. El objetivo principal que persigue es el de componer música a través de un computador, ya sea de forma asistida por un humano o autónoma. La ejecución de esta tarea ha sido posible debido a distintas implementaciones que han ido desarrollándose a lo largo del tiempo; no obstante, con la

aparición de las redes neuronales, este campo ha incrementado su popularidad, y grandes industrias han invertido en proyectos sobre el mismo. Uno de los mayores ejemplos que podemos encontrar es el proyecto **Magenta** [2] de Google, del cual hablaremos más adelante.

A pesar de haber nacido de forma posterior a la composición artificial, la recuperación de información musical (Music information retrieval), a partir de ahora **MIR** por sus siglas en inglés, ha evolucionado en paralelo a esta ya que utiliza tecnologías similares en muchas de sus aplicaciones. En sí las técnicas MIR se basan en la obtención de características directamente desde un audio. Dentro de sus aplicaciones podemos encontrar técnicas tales como el reconocimiento de voz, la clasificación musical por géneros o el etiquetado musical.

Tal y como se ha mencionado antes, la música sufre constantes cambios y principalmente estos vienen dados por la época y el contexto en el que se desarrolla, por lo que es lógico pensar que la aplicación de todas estas técnicas a la composición musical sea uno de los siguientes pasos a dar en la historia de la música. A pesar de la "facilidad" con la que un usuario puede generar música artificial, ésta música generada tiene ciertas limitaciones que la separa de la música real. Al ser esta diferencia uno de los mayores obstáculos existentes en la composición artificial es importante estudiar cómo superar estas limitaciones.

Este trabajo se centra en el análisis y estudio de las diferencias existentes entre la música real y la música artificial. Para llevar a cabo esto se han utilizado diferentes técnicas de composición artificial así como técnicas MIR a partir de las cuales estudiaremos distintos aspectos de ambos tipos de música. El análisis girará en torno a la comparación mediante técnicas de clasificación de género, etiquetado musical y los resultados de una encuesta pública propuesta. Todos los procedimientos y técnicas usadas serán descritas a fondo posteriormente.

1.2. Objetivos

Para el desarrollo de este trabajo establecemos los siguientes objetivos.

O-1.– Creación de un conjunto de canciones de géneros populares (Pop, Rock, RnB...) en formato MIDI que sea adaptable a distintos modelos, tanto para composición artificial como para las técnicas MIR.

O-2.– Generación de composiciones artificiales a partir modelos que proporcionen diferentes salidas. De este modo seremos capaces de analizar el problema desde distintos puntos de vista.

O-3.– Encuesta de evaluación musical. Esta encuesta estará formada por composiciones artificiales que serán evaluadas por todo aquel que realice ésta.

O-4.– Obtención de características para el conjunto del **O-1** y el conjunto de composiciones artificiales a través de técnicas de clasificación de género y de etiquetado musical.

O-5.– Encontrar a partir de los resultados obtenidos en el **O-3** y **O-4** las diferencias y semejanzas existentes entre los distintos conjuntos de datos. Éstas pueden ser independientes de cada modelo o no, por lo que se considera este objetivo el más importante e interesante en el desarrollo del trabajo.

1.3. Organización de la memoria

Esta memoria consta de los siguientes capítulos:

- **Introducción:** Introducción del trabajo en la que se trata acerca de la motivación que envuelve a este y de los objetivos a llevar a cabo.
- **Estado del arte:** Resumen de la historia de la composición artificial y las técnicas MIR a través de los artículos que más han definido estas metodologías. Finalmente se tratará el estado actual de estas y sus casos de uso más habituales.
- **Técnologías usadas:** Descripción de cada componente utilizado en la fase experimental.
- **Fase experimental:** Desarrollo del problema y de las fases que comprenden éste, así como todas las decisiones llevadas a cabo y problemas encontrados.
- **Análisis de los resultados:** Análisis de resultados de forma individual y conjunta. Cada metodología de composición artificial será estudiada mediante las técnicas MIR ya mencionadas y posteriormente se analizarán los resultados de forma general a todos los modelos.
- **Discusión:** Opinión personal del autor respecto a como estas nuevas metodologías afectan a la música.
- **Conclusiones:** Conclusiones obtenidas a partir del análisis de los resultados.

2

Estado del arte

En este capítulo trataremos las bases sobre las que se establece el proyecto desde un punto de vista histórico y teórico. Referenciaremos las publicaciones más relevantes y los casos de uso más populares en la actualidad. Para ello, vamos a separar los dos conceptos principales a tratar, composifición artificial y recuperación de información musical. Una vez tratados estos, pasaremos a hablar de los modelos utilizados en el presente trabajo, **basados en redes neuronales articificiales** y **deep learning**.

2.1. Composición artificial

Llamamos **composición artificial** a toda composición musical que ha sido generada a través de inteligencia artificial, sin embargo, para comprender realmente la composición artificial debemos comprender primero qué es la composición algorítmica.

La composición algorítmica se basa en la utilización de algoritmos para componer música. Un algoritmo es un secuencia de reglas (instrucciones, operaciones) ordenadas y finitas que pretenden solucionar un problema. En este caso, los problemas a resolver son las distintas piezas que componen una pieza musical. Su origen no está muy claro, sin embargo uno de los primeros casos de uso más conocidos de ésta es el juego de dados musical popularizado por Mozart [3] entre la segunda mitad del siglo XVIII y la primera mitad del XIX. En él, primero se atribuía a cada cara de un dado un pasaje musical, después se realizaban varias tiradas y se ordenaban una detrás de otra dando lugar de esta manera a distintas composiciones.

Con el constante desarrollo de la tecnologia la composición algorítmica se unió a la inteligencia artificial dándo lugar a distintos sistemas que tratan este problema.

2.1.1. Modelos algorítmicos

G. Papadopoulos y G. Wiggins proponen en [4] una clasificación de modelos algorítmicos que se convirtió en estado del arte. Ejemplo de esto es el artículo de JD. Fernández y F. Vico [5] de 2013 en

el que usan esta división como base pero actualizándola a las tecnologías más recientes.

En este artículo JD. Fernández y F. Vico proponen los siguientes métodos:

- **Gramáticas:** Las gramáticas formales son un conjunto de reglas que permiten utilizar un lenguaje de alto nivel de forma secuencial a partir del cual formar palabras. Estas palabras representan los resultados del problema que aborda la gramática en cuestión. En el caso de la música, las palabras representarían las distintas partes de la composición o la propia composición. Encontramos distintos usos de las gramáticas de cara a la composición musical, como por ejemplo Lindenmayer Systems o L-Systems [6] o un uso combinado de las gramáticas y los algoritmos evolutivos.
- **Sistemas basados en el conocimiento:** Atendiendo a la definición que encontramos en [7], los sistemas basados en conocimiento o **KBS** (Knowledge Based Systems) por sus siglas en inglés, son sistemas que usan la IA para resolver problemas. Estos sistemas incorporan un conjunto de conocimientos especializados con utilidades diseñadas para facilitar la recuperación de estos a partir de consultas específicas. Estos también pueden ser transferidos de un ámbito de conocimientos a otro. El conocimiento musical puede interpretarse como un conjunto de reglas que podemos utilizar de distintas formas para crear composiciones, es por esto que de cara a la composición algorítmica, los KBS son usados junto a sistemas basados en reglas.
- **Cadenas de Markov:** Las cadenas de markov parten del concepto en el que se basan los procesos estocásticos discretos. Estos procesos comprenden un conjunto estados a partir de los cuales se tránsita. En cada estado se depende solamente del estado anterior por lo que denominamos éstos como procesos faltos de memoria (Propiedad de Markov). Estos métodos han estado presentes en la composición artificial durante mucho tiempo, ejemplo de esto es el artículo de Charles Ames de 1989 que proponía la utilización de procesos de markov enfocado a modelos de composición [8].
- **Redes neuronales artificiales:** Las redes neuronales artificiales son modelos computacionales basados en el comportamiento del cerebro humano y las conexiones entre las neuronas que éste contiene. Esta representación del cerebro se realiza mediante un conjunto de neuronas organizadas en capas y una serie de conexiones entre estas. Podemos dividir las capas en tres tipos:
    - **Capa de entrada:** Esta capa se encarga de recibir los datos de entrada de la red neuronal y de enviarlos a las capas ocultas.
    - **Capas ocultas:** En estas capas es dónde realmente se retiene la información que la red aprende y se encargan de procesar los datos y enviarlos a la capa de salida.
    - **Capa de salida:** Por último, la capa de salida se encarga de aplicar una función (generalmente no lineal) sobre los datos recibidos de la última capa oculta que determinará el resultado de los datos procesados.

  Las neuronas por su parte cuentan con funciones de activación que, con base en los datos recibidos de las capas anteriores transmiten o no los datos a las siguientes capas. Podemos encontrar una comparativa interesante sobre estas funciones de activación en [9]
- **Métodos evolutivos o basados en población:** Los métodos evolutivos se basan en la utilización de algoritmos genéticos. Éstos parten de un conjunto de individuos que son evaluados, seleccionados y alterados con variaciones en su composición. Este proceso se repite de forma iterativa hasta encontrar las soluciones óptimas. En lo referente a la composición musical, este campo al igual que las cadenas de Markov ha sido investigado desde hace mucho tiempo. Uno de los primeros artículos que aplica algoritmos genéticos a la composición musical es el desarrollado por Andrew Horner y David E. Goldberg [9]. Los autores parten de la idea principal de que estos métodos han sido utilizados en otros ámbitos como por ejemplo la medicina o la clasificación de procesos productivos.
- **Autosimilitud y autómata celular:** La autosimilitud es la propiedad de un objeto en la que el propio objeto en sí es muy similar o exacto a una de las partes que conforman este. Esta propiedad es una de las características

que definen los fractales. J. Hsü y A. Hsü defienden en [10] que la autosimilitud es una de las propiedades de la música clásica, encontrando patrones y propiedades de ésta en composiciones de Mozart o Beethoven. Los autómatas celulares [11] son otra de las alternativas a la hora de componer música, siendo capaces de generar también patrones fractales. Ambas metodologías consiguen reproducir patrones existentes en las composiciones musicales, sin embargo, una de sus limitaciones es que requiere de un trabajo humano posterior para que estas se asemejen a una composición real, por lo que no parecen ser más que una fuente de inspiración.

En [5] se encuentra una explicación más detallada y ejemplos de todos estos modelos por separado.

2.2. Recuperación de información musical

La recuperación de información musical es el conjunto de técnicas que nos permite extraer información de la música. Estas técnicas nacen de la necesidad de automatizar tareas que si son realizadas a mano requieren un gran esfuerzo. Ejemplos de estas tareas son:

- Detección de canción versionada
- Reconocimiento de género
- Transcripción
- Recomendación
- Separación de instrumentos
- Estimación de tempo
- Detección de clave de la composición

La mayoría de estas técnicas están basadas en la extracción de características directamente desde la señal de audio. Algunas de ellas cuentan también con un trabajo previo de procesado que facilita posteriormente la automatización.

Existen distintos tipos de características extraíbles, sin embargo las más utilizadas son aquellas que nos permiten almacenar información sobre la intensidad del espectro de frecuencias a lo largo del tiempo. Esta información obtenida también puede ser interpretada mediante un espectrograma por lo que obtendríamos una representación visual de los datos obtenidos.

Una de las limitaciones de estas características es la necesidad de representar un conjunto de frecuencias muy grande a lo largo del tiempo. El espectro audible del oído humano figura entre los 20 Hz y los 20kHz por lo que tendríamos que almacenar información sobre cada una de estas frecuencias a lo largo de la duración de la señal de audio a analizar. A pesar de que la duración de los fragmentos a usar no suele superar los 4 segundos esto ya supone una gran cantidad de datos a almacenar.

De esta limitación nacen distintas representaciones de las señales de audio (tiempo-frecuencia) que permiten almacenar esta misma información pero en un formato más compacto. En [12] encontramos una recopilación de las representaciones de datos de audio más utilizadas en técnicas MIR actualmente. Explicaremos una a una las características y propiedades que las definen.

2.2.1. Transformada de Fourier de tiempo reducido (STFT)

La transformada de Fourier de tiempo reducido o **STFT** (Short Time Fourier Transform) por sus siglas en inglés, está basada en la transformada de Fourier, una transformación matemática que nos permite calcular la contribución de cada valor de frecuencia a la formación de la señal a lo largo del tiempo.

Una de las propiedades de la STFT es su capacidad de ser invertible, pudiendo obtener a partir de la representación frecuencias-tiempo la señal original. La fórmula a partir de la cual obtenemos la STFT es la siguiente:

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n-m)e^{-j\omega n} \tag{2.1}$$

Dónde m representa tiempo, ω frecuencia, $x(n)$ los fragmentos de la señal de audio a procesar y $w(n-m)$ una función de ventana.

A partir de esta transformación obtenemos una representación de la señal de audio en sus frecuencias lineales centrales, sin embargo, éstas son distintas a las del espectro audible humano y no son frecuencias utilizadas con algún tipo de motivación musical, por lo que no son tan útiles como las obtenidas a partir de otros métodos.

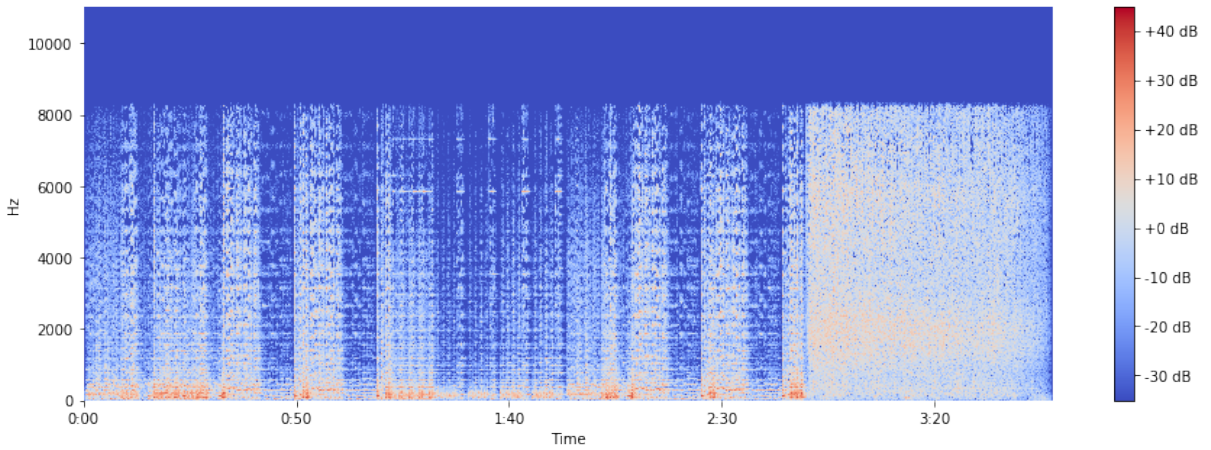


Figura 2.1: Transformada de Fourier en tiempo reducido: espectrograma

Las ventajas que presenta esta representación son su rapidez de cálculo ($O(log(n))$) frente a otras representaciones tiempo-frecuencia y su posibilidad de inversión, siendo ésta última muy útil en procesos de sonificación de características aprendidas [13] y separación de instrumentos [14].

Todas las figuras pertenecientes a este apartado han sido extraídas de la página web https://musicinformationretrieval.com/stft.html.

2.2.2. Espectrograma de Mel (MFCC)

El espectrograma de Mel es una representación de los Coeficientes Cepstrales en las Frecuencias de Mel o **MFCCs** (Mel Frequency Cepstral Coefficients) por sus siglas en inglés. Estos coeficientes fueron utilizados inicialmente en técnicas de reconocimiento de voz, convirtiéndose en estado de arte desde entonces. Posteriormente, en el año 2000, B Logan et al. [15] aplicaron estos coeficientes a distintas técnicas MIR, obteniendo exitosos resultados que convertirían a estos coeficientes en una referencia en el ámbito MIR.

El proceso de obtención de los MFCCs parte de la idea de los STFT. El procedimiento de obtención es el siguiente:

1.– Dividimos la señal en distintos fragmentos. (Fragmentos de $\pm$ 20ms)

2.– Aplicamos la transformación de fourier discreta. (Obtenemos los STFT)

3.– Obtenemos el logaritmo de la amplitud del espectrograma del paso 2. Descartamos la información de fase.

4.– Aplicación del escalado y suavizado de Mel.

5.– Aplicación de la transformación de coseno discreta.

Aplicando estos pasos, obtenemos una representación de la intensidad de las frecuencias a través de la escala de Mel. Según [12] y [16], una forma adecuada de aplicar esta escala es a través de la siguiente fórmula:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.2}$$

Dónde f representa la frecuencia (en herzios) a transformar.

Al igual que en el caso de los STFT podemos representar estos datos en un espectrograma, sin embargo, debido a que solo almacenamos la información relativa a la intensidad de las frecuencias perdemos la propiedad de inversión de la señal.

Una de las ventajas a la hora de usar los MFCCs es la similitud de sus frecuencias respecto a las audibles por el oído humano, resultando éstos por lo tanto muy útiles en distintas tareas relativas a las técnicas MIR. Ejemplos de esto son las tareas de clasificación por géneros [17] o de etiquetado musical [18].

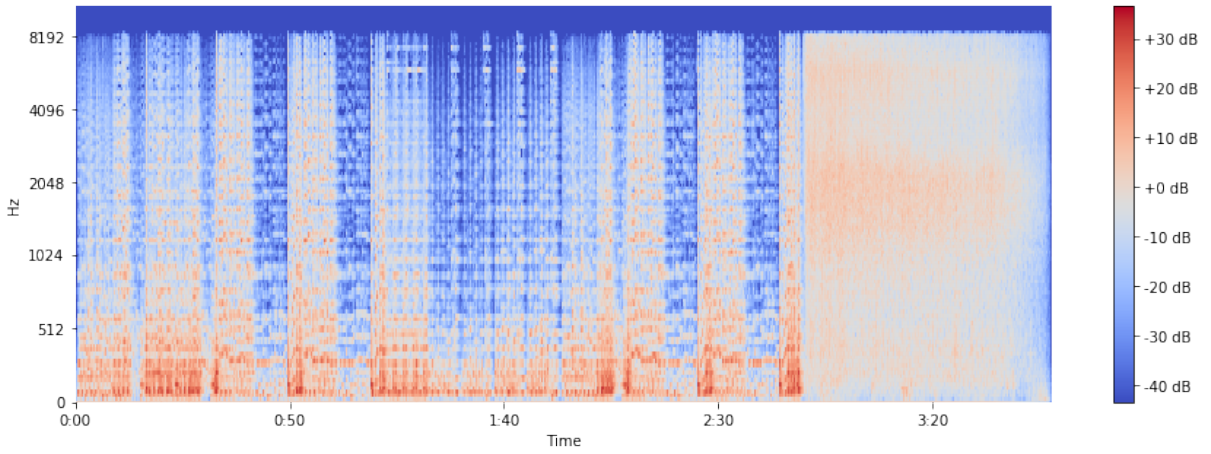


Figura 2.2: Espectrograma de Mel

2.2.3. Transformación Q constante (CQT)

La Tranformación Q constante o **CQT** (Constant Q Transform) por sus siglas en inglés, realiza una representación tiempo-frecuencia mediante un eje de frecuencias logarítmico y a través de un conjunto de filtros que dividen éste. A partir de éstos se obtienen unas frecuencias centrales con una separación entre ellas similar a la existente entre las notas pertenecientes a las escalas musicales occidentales.

Las frecuencias centrales hacen referencia a aquella que se encuentra entre la frecuencia más baja y más alta de un filtro aplicado a una señal. La fórmula para obtenerlas es la siguiente:

$$f_c(k_{lf}) = f_{min} \cdot 2^{k_{lf}/\beta} \tag{2.3}$$

Dónde f_{min} hace referencia a la frecuencia central del filtro más bajo, k_{lf} hace referencia al filtro a partir del cual obtener la frecuencia central y β hace referencia al número de filtros por octava.

Los filtros que dividen el eje de frecuencias dependen de varios factores y obtienen su ancho espectral a partir de la siguiente fórmula:

$$\delta f_k = 2^{1/\beta} \cdot \delta f_{k-1} = (2^{1/\beta})^k \cdot \delta f_{min} \tag{2.4}$$

Dónde δf_k hace referencia al ancho espectral del filtro, f_{min} hace referencia a la frecuencia central del filtro más bajo y β hace referencia al número de filtros por óctava.

Encontramos en [19] este proceso de transformación de forma detallada además de varios espectrogramas de instrumentos transformados. Sus aplicaciones más comunes son las relacionadas con el reconocimiento de acordes [20] y la transcripción musical [21].

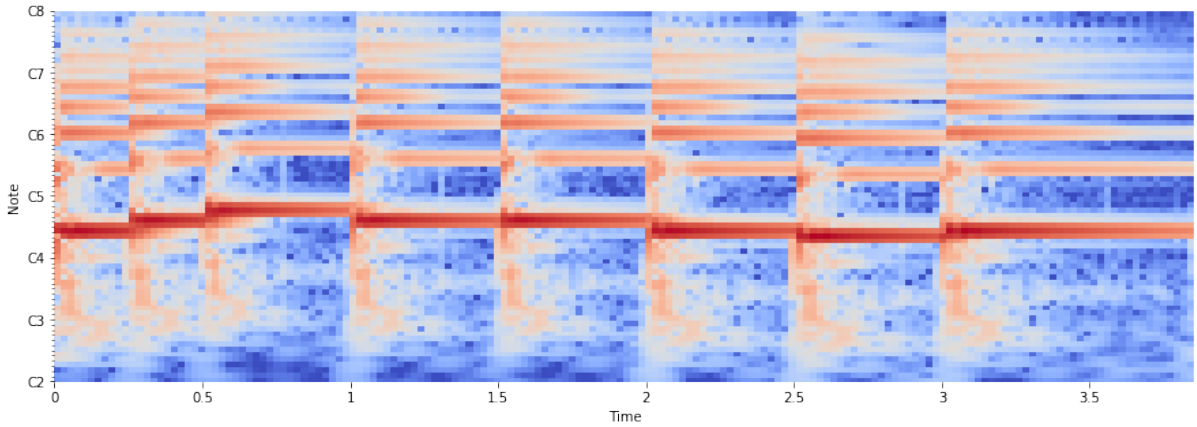


Figura 2.3: Espectrograma CQT (Eje frecuencias transformado a notas musicales. CX representa el comienzo de una escala.)

2.2.4. Cromagrama

El cromagrama es la representación de una de las escalas en las que está dividido un eje de frecuencias logarítmico. Suele estar asociado al CQT pero puede utilizarse a partir de cualquier espectro de frecuencias logarítmico. Podemos obtener este cromagrama a partir de la siguiente fórmula:

$$C_f(b) = \sum_{z=0}^{Z-1} |X_{lf}(b + z\beta)| \tag{2.5}$$

Dónde Z representa el número de escalas (octavas), X_{lf} el espectro de frecuencias logarítmico, b es el índice del tono (Pitch class) (Por cada escala tenemos un tono repetido, es por esto que existe un índice para cada tono) y β indica el número de filtros por óctava.

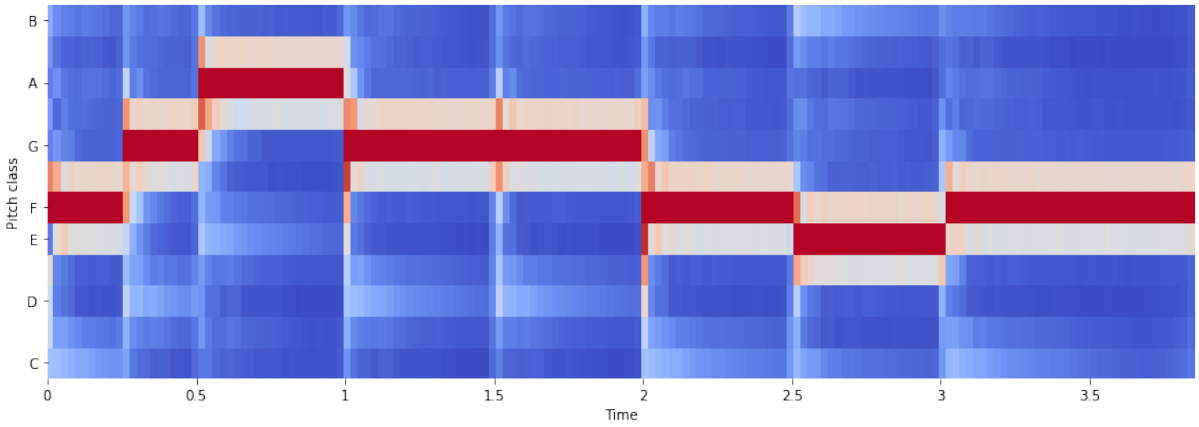


Figura 2.4: Cromagrama CQT (Eje frecuencias transformado a notas musicales en una sola escala.)

2.3. Redes neuronales artificiales

Como ya hemos mencionado en el apartado 1.1, una de las metodologías más populares y utilizadas en la actualidad son las redes neuronales artificiales o **NN** (Neural networks) por sus siglas en inglés. Estos modelos se emplean en el ámbito del **aprendizaje automático** (Machine learning) [22] y han sido utilizados principalmente desde su descubrimiento en tareas de **aprendizaje supervisado**, sin embargo, con el avance tecnológico y la aparación del **Big data** [23] ha sido necesario avanzar hacia el **aprendizaje no supervisado**. Surge así el concepto de **Aprendizaje profundo** (Deep Learning).

2.3.1. Aprendizaje Profundo (Deep Learning)

Podemos definir el **Aprendizaje Profundo** [24] como un conjunto de técnicas de aprendizaje automático aplicadas sobre las NN que nos permiten realizar tareas de aprendizaje supervisado y no supervisado, tanto con pequeñas como con grandes cantidades de datos.

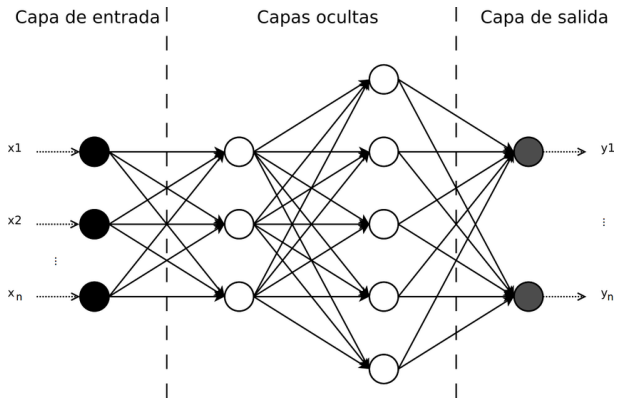


Figura 2.5: Ejemplo de red neuronal con 2 capas ocultas [1]

Partiendo del concepto de NN explicado en el apartado 2.1.1, podemos observar la estructura de una red neuronal en la figura 2.5. Existen multitud de modelos de NN, sin embargo, debido al alcance del trabajo solo explicaremos aquellos que se han usado a lo largo del desarrollo de este.

2.3.2. Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes o **RNN** (Recurrent Neural Networks) por sus siglas en inglés, se desarrollaron en los años 80 y desde entonces se han utilizado en múltiples ámbitos del aprendizaje automático. Sin embargo, debido a la carga computacional de éstas, su potencial no ha podido ser aprovechado hasta la época actual.

[1] Extraído de: https://www.researchgate.net/figure/Red-neuronal-artificial-de-cuatro-capas_fig1_323985249

La idea principal de la que parten estas redes es la de almacenar información a lo largo del tiempo, dotando a la red en cierto modo de "memoria". Para llevar esto a cabo utilizan las denominadas células de memoria (memory cells), que, a diferencia de las neuronas de una NN normal, generan además de su salida, una salida más que se mandan a ellas mismas. Podemos observar esta diferencia en la figura 2.6.

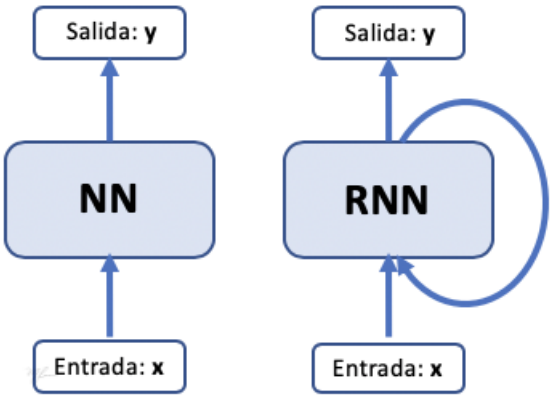


Figura 2.6: Neurona normal frente a neurona perteneciente a RNN [2]

La información que se retiene de la salida se transmite a la misma neurona en el siguiente paso dado dentro del entrenamiento de la RNN, siendo éste el mecanismo que permite dotar de memoria a la red. De este modo, las entradas de las neuronas de una RNN son la propia entrada de la neurona más el valor de la salida de la neurona en el instante anterior.

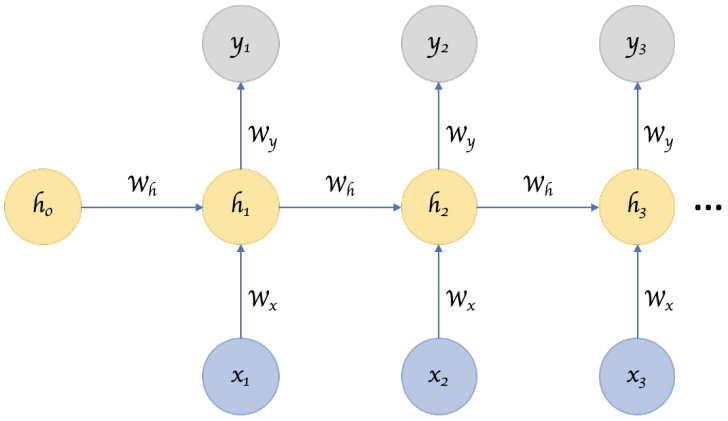


Figura 2.7: Neurona RNN a lo largo del tiempo. [3]

A pesar de estar “dotada de memoria” ésta red sufre una limitación común en muchas NN y viene dada por una de las técnicas usadas durante el entrenamiento. Ésta técnica es la **propagación hacia atrás** (back-propagation), un mecanismo que permite ajustar los pesos de una NN en función del error

[2] Extraído de: http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/RNN_LTSM/introduccion_rnn.html
[3] Extraído de: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

obtenido en la salida. Decimos que una red sufre de **desvanecimiento de gradiente** [25] (Vanishing gradient) cuando ésta propagación hacia atrás no se puede realizar correctamente debido a que los valores del gradiente son muy pequeños. Este problema deriva en pesos que no se consiguen ajustar correctamente y por ende, tiempos de entrenamiento muy largos y costosos computacionalmente.

Una de las soluciones para este problema fue la introducción de las neuronas con memoria a largo plazo o **LSTM** (Long short-term memory) por sus siglas en inglés.

Neuronas con memoria a largo plazo (LSTM)

Las neuronas LSTM también son consideradas neuronas con memoria, sin embargo, éstas tienen la capacidad de retener información en un periodo de tiempo mucho más largo que las de una RNN y de elegir qué información almacenar. Para comprender ésto mejor, revisamos brevemente la estructura y funcionamiento de una neurona LSTM.

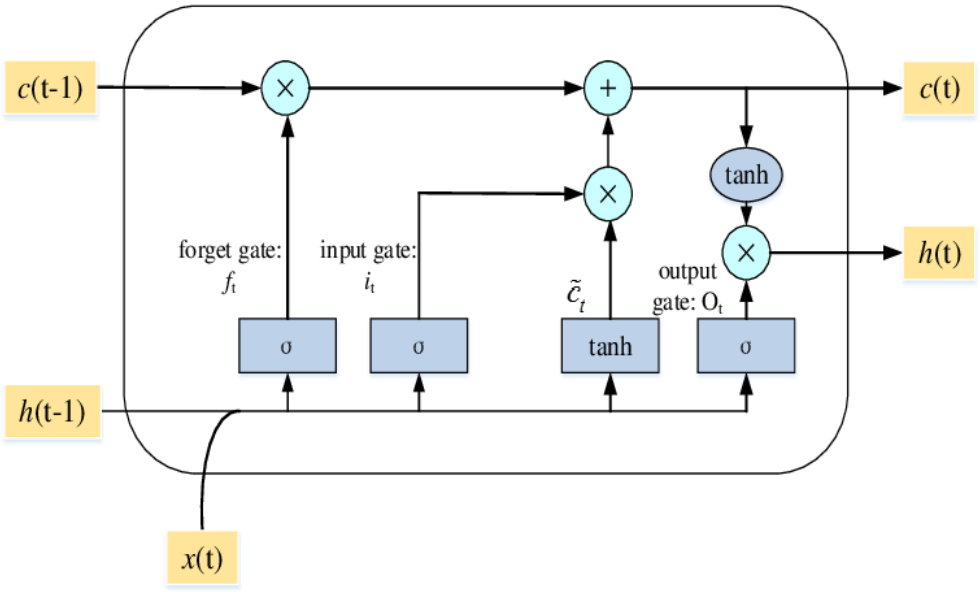


Figura 2.8: Estructura neurona LSTM. [4]

Como podemos observar en la figura 2.8, éstas neuronas cuentan con una puerta para olvidar (forget gate), una puerta de entrada (input gate) y una puerta de salida (output gate). Es gracias a la acción conjunta de éstas tres puertas y a una serie de operaciones que la neurona es capaz de decidir si modificar o eliminar su información.

La **puerta de entrada** se encarga de filtrar los valores que se van a utilizar para modificar la memoria. Ésto se lleva a cabo mediante una función sigmoidal y una función tangente hiperbólica sobre los datos de entrada.

[4] Extraído de: https://www.researchgate.net/figure/The-structure-of-the-LSTM-unit_fig2_331421650

La **puerta de olvido** se encarga de eliminar la información que ya no es relevante dentro de la neurona. Ésto lo realiza mediante una función sigmoidal sobre el anterior estado de la neurona y los nuevos datos de entrada.

La **puerta de salida** se encarga de calcular el valor de salida (El estado de la celda). Ésto lo realiza mediante una función sigmoidal sobre los datos de entrada y una función tangente hiperbólica sobre la información almacenada en la memoria de la neurona.

En la actualidad la mayoría de las implementaciones de RNN se llevan a cabo con neuronas LSTM debido a su mayor eficacia de cara al entrenamiento de éstas. Entre sus usos más comunes encontramos aquellos problemas que requieren de un procesamiento de datos secuencial. Ejemplos de esto son, reconocimiento de voz [26], *machine reading* [27] o clasificación de imágenes [28].

2.3.3. Redes neuronales convolucionales (CNN)

Del mismo modo que las RNN, las redes neuronales convolucionales o **CNN** (Convolutional Neural Network) por sus siglas en inglés, son un modelo que ha sido utilizado a lo largo de le época moderna de los computadores, sin embargo, su verdadero potencial no se ha desarrollado hasta la actualidad. Éstas redes deben su popularidad a la efectividad con la que resuelven problemas de clasificación de imágenes [29], reconocimiento en imágenes/vídeos [30] [31], sistemas de recomendación [32] o problemas de procesado de lenguaje natural [33] entre otros.

La particularidad de éstas redes neuronales yace en su estructura ya que la entrada de las capas ocultas pasa a estar formada por un conjunto de **capas convolucionales** y **capas de *pooling***. Gracias a éstas, las CNN son capaces de aprender qué información es relevante mediante el entrenamiento, actuando estas capas como filtros. Podemos apreciar esto en la figura 2.9

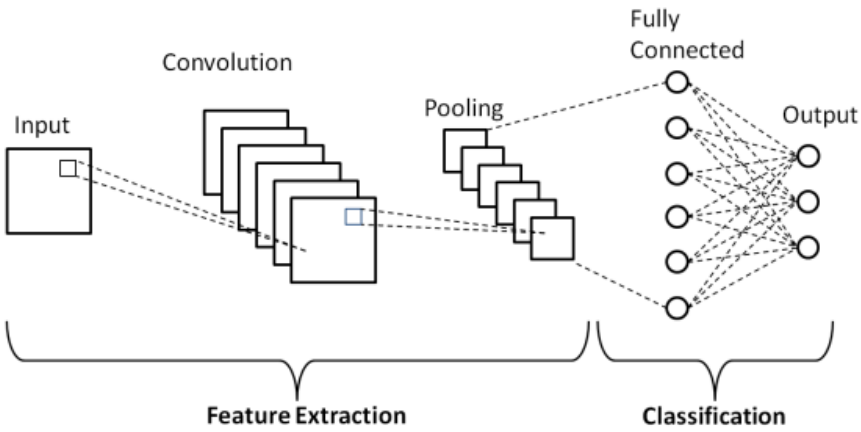


Figura 2.9: Estructura de una red neuronal convolucional (CNN). [5]

[5] Extraído de: https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909

La función de las **capas convolucionales** es extraer características de los datos de entrada a traves de un filtro o kernel que se aplica a éstos. Una vez éste se ha aplicado obtenemos nuevos datos con una dimensión menor que se enviarán a las capas de pooling. En la figura 2.10 observamos los datos de entrada $\mathbf{I}$, el filtro o kernel $\mathbf{K}$ y los datos obtenidos al aplicar la convolución $\mathbf{I} * \mathbf{K}$.

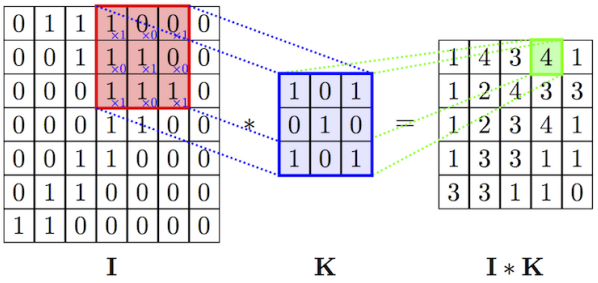


Figura 2.10: Proceso de convolución. [6]

Las **capas de pooling** realizan una función similar a las capas convolucionales, sin embargo éstas se encargan de reducir la dimensión de los datos de entrada, manteniendo aquellos que son más relevantes. Ésto es lo que se conoce como ***max. pooling***. Una vez los datos han pasado por éste proceso pueden ser mandados a una nueva capa convolucional o directamente a las capas ocultas dónde serán clasificados.

A parte de la convolución y el *pooling* existen otras técnicas como el ***padding***, el ***stride*** o el ***stacking*** que nos permiten ajustar el proceso de convolución de cara a las dimensiones de los datos obtenidos de éste. En [34] encontramos una explicación detallada sobre las CNN y los distintos procesos llevados a cabo en sus capas de convolución y pooling.

2.3.4. Redes adversarias generativas (GAN)

Las redes adversarias generativas o **GAN** (Generative Adversarial Networks) por sus siglas en inglés, fueron propuestas en el año 2014 [35] y plantean una estructura compuesta por dos redes neuronales profundas, una **red generativa** o generador y una **red discriminatoria** o discriminador. Podemos ver su estructura en la figura 2.11. Sus principales aplicaciones son las enfocadas a la genéracion y análisis de multimedia. Podemos ver ejemplos ésto en: imágenes [36] [37], música [38] y vídeos [39].

El objetivo principal de éstas redes es conseguir que el generador sea capaz de generar información realista. Comentaremos superficialmente el funcionamiento de las GAN. Como en cualquier otra red necesitamos unos datos de entrenamiento, en este caso los denominaremos *datos reales*.

Comenzamos explicando el funcionamiento del discrimador. Esta red recibirá tanto datos reales co-

[6] Extraído de: https://blog.francium.tech/machine-learning-convolution-for-image-processing-42623c8dbec0

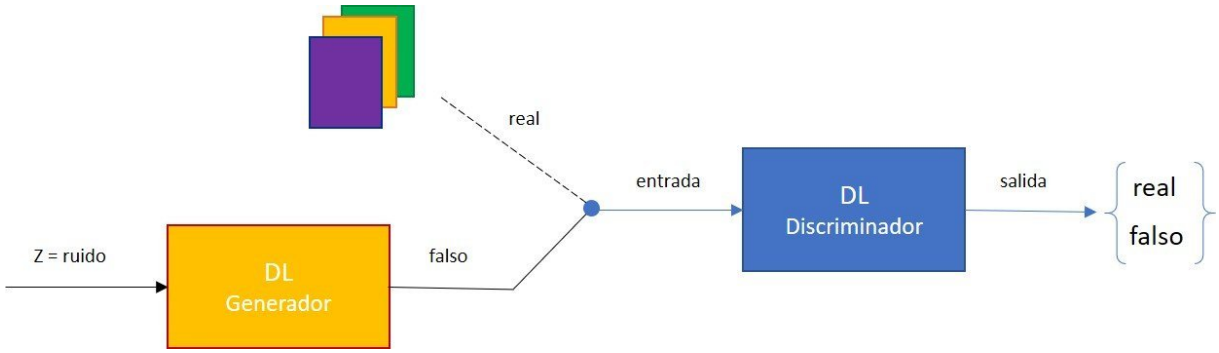


Figura 2.11: Estructura de una red adversaria generativa. [7]

mo datos falsos, los cuales serán suministrados por el generador. Su función es ser capaz de distinguir éstos, por lo que a lo largo del entrenamiento será capaz de realizar esto con mayor precisión. Como hemos mencionado antes, el generador suministra información falsa, la cual está basada en una fuente de ruido. Su función es aprender a "engañar" al discriminador ajustando poco a poco la información que le suministra. A lo largo del entrenamiento la red generadora proporcionará al discriminador datos cada vez más similares a los del entrenamiento, sin embargo, al partir éstos de una fuente de ruido nunca serán iguales. Es gracias a este compartamiento que la red generadora una vez haya finalizado el entrenamiento será capaz de generar resultados realistas.

Con todos los conceptos relativos al estado del arte introducidos, damos paso al siguiente capítulo.

[7] Extraído de: https://www.iartificial.net/redes-neuronales-generativas-adversarias-gans/

3

Tecnologías utilizadas

En esta sección introduciremos las tecnologías usadas para el desarrollo del proyecto de forma individual, haciendo referencia a los conceptos sobre los que se basan y los modelos utilizados dentro de éstas.

Existen múltiples razones por las cuales estas tecnologías han sido seleccionadas para el proyecto. De cara a las herramientas de generación musical las razones principales que han determinado su selección son la extensa documentación de la que disponen y su adaptabilidad al formato **MIDI** [40] en la gran mayoría de sus configuraciones. En cuanto a las técnicas MIR, se han seleccionado modelos que estén pre-entrenados sobre conjuntos de datos comúnmente usados en tareas de recuperación de información y que aporten información de alto nivel en sus resultados, permitiendo esto realizar un análisis más generalizable en cuanto a composiciones musicales se refiere.

3.1. Magenta

Como ya mencionamos en el apartado 1.1, **Magenta** es un proyecto de investigación de código abierto desarrollado por el equipo *Google Brain*. Su enfoque principal es la creación de arte a través de técnicas de aprendizaje automático, tal y como mencionan en [2]:

"WHAT IS MAGENTA?
An open source research project exploring the role of machine learning as a tool in the creative process."

Para llevar a cabo estos procesos creativos desarrollan herramientas y modelos que utilizan tecnología basada en Deep Learning y algoritmos de aprendizaje reforzado. El objetivo de estas herramientas es ser un soporte para los artistas y no un reemplazo. Todas éstas herramientas desarrolladas son creadas a partir de la biblioteca de código abierto desarrollada por Google, **Tensorflow** [41]. La aparición de ésta supuso una revolución gracias a su fácil adaptabilidad y usabilidad, disponiendo de APIs en distintos lenguajes de programación, como Python, JavaScript , C++, etc.

Dependiendo del enfoque artístico y lenguaje de programación que usemos, Magenta nos permite usar distintos modelos. Podemos encontrarlos en los repositorios de GitHub, accesibles desde su página web principal.

Para el desarrollo de este trabajo se ha utilizado la API de python con 3 modelos distintos enfocados a la generación musical. A pesar de ser 3 modelos diferentes, una de las ventajas de Magenta es que nos permite transformar nuestros datos de entrenamiento a un formato que aceptan la gran mayoría de sus modelos. Éste formato es **NoteSequences** y nos permite convertir una colección de MIDIs que tengamos almacenadas en una carpeta en un solo fichero. El proceso de conversión lo podemos encontrar en el siguiente enlace: https://github.com/magenta/magenta/blob/master/magenta/scripts/README.md. Una vez hemos transformado nuestros MIDIs cada modelo nos proporcionará los comandos necesarios para su adaptación a éste. Todos estos modelos están basados en RNN con neuronas LSTM, ya que, como comentamos en el apartado 2.3.2, se adaptan correctamente a problemas que requieren tratar datos de forma secuencial, como es el caso de la música.

3.1.1. Melody RNN

El modelo Melody RNN está enfocado a la generación de melodías monofónicas, esto quiere decir que en sus generaciones (composiciones) nunca observaremos que dos notas o más que suenen en el mismo instante. Observamos este concepto en la figura 3.1

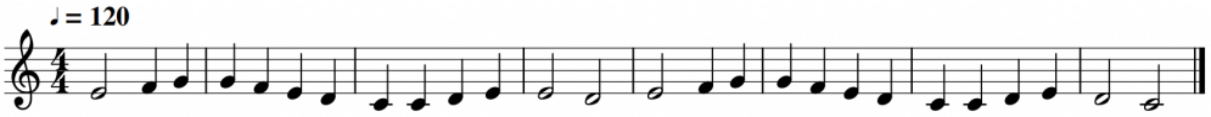


Figura 3.1: Ejemplo de melodía monofónica (Compás 4/4 y tempo 120).

Existen diferentes configuraciones, siendo cada una más compleja (y costosa computacionalmente) que la anterior. Éstas son:

- **Basic:** *"Esta configuración actúa como base para la generación de melodías con un modelo LSTM. Utiliza una codificación one-hot para representar las melodías extraídas como entrada al LSTM. Para el entrenamiento, todos los ejemplos se transponen al rango de tono MIDI [48, 84]. Las salidas también se generarán en ese rango."*
- **Mono:** *"Mientras que **basic rnn** se entrena transponiendo todas las entradas a un rango pequeño, **mono rnn** es capaz de utilizar los 128 tonos MIDI completos."*
- **Lookback:** *"Lookback RNN introduce entradas y etiquetas personalizadas. Las entradas personalizadas permiten al modelo reconocer más fácilmente los patrones que se producen a lo largo de 1 y 2 compases... Las etiquetas personalizadas reducen la cantidad de información que el estado de la celda del RNN tiene que recordar, permitiendo al modelo repetir más fácilmente los eventos de hace 1 y 2 compases. Esto da como resultado melodías con mayor coherencia y estructura más musical."*
- **Attention:** *"Esta configuración introduce el uso de la atención. Ésta permite al modelo acceder a la información del pasado sin tener que almacenarla en el estado de la célula RNN, aprendiendo más fácilmente las dependencias a largo plazo y dándo como resultado melodías que mantienen su coherencia durante más tiempo."*

Encontramos en [42] un artículo propio del blog de Magenta en el que se explican las ventajas de las configuraciones *attention rnn* y *lookback rnn* sobre el resto de configuraciones, acompañado de ejemplos obtenidos por ellos mismos.

3.1.2. Polyphony RNN

El modelo Polyphony RNN está enfocado a la generación de melodías polifónicas. Al contrario que en las melodías monofónicas, éstas nos permiten escuchar varias notas simultáneamente. Observamos este concepto en la figura 3.2

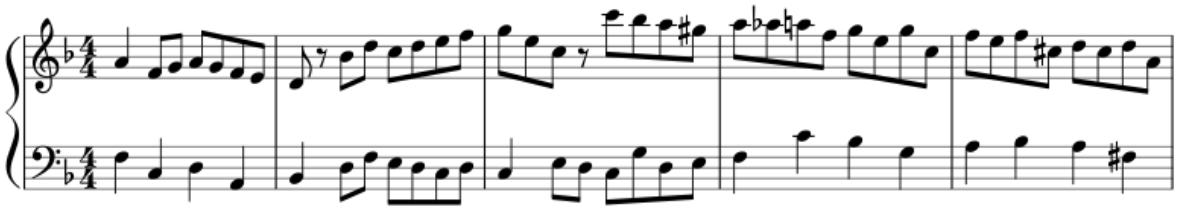


Figura 3.2: Ejemplo de melodía polifónica (Compás 4/4 en ambas voces y tempo desconocido).

Éste modelo encuentra su inspiración en el doodle ***BachBot*** [43], *"...modelamos la polifonía como una secuencia única de eventos de notas con símbolos especiales de INICIO, FINAL DE PASO y FINAL. Dentro de un paso, las notas se ordenan por tono en orden descendente. Por ejemplo, utilizando la resolución de cuantificación predeterminada de 4 pasos por negra, una secuencia que contenga sólo un acorde de do mayor con una duración de una negra..."*, se representaría como en el cuadro de texto 3.1

```
START
NEW_NOTE, 67
NEW_NOTE, 64
NEW_NOTE, 60
STEP_END # Fin paso 1
CONTINUED_NOTE, 67
CONTINUED_NOTE, 64
CONTINUED_NOTE, 60
STEP_END # Fin paso 2
# Repetimos dos veces más el paso 2
END
```

Cuadro 3.1: Representación de un acorde do mayor con duración de negra en Polyphony RNN

3.1.3. Performance RNN

El modelo Performance RNN también está enfocado a la generación de melodías polifónicas, sin embargo, se diferencia del anterior modelo en la capacidad de introducir eventos en las notas de sus generaciones. Los eventos que puede tener cada nota son los siguientes:

- **NOTE_ON(tono):** Comienza una nota en el tono indicado.
- **NOTE_OFF(tono):** Finaliza una nota en el tono indicado.
- **TIME_SHIFT(cantidad):** Desplaza el tiempo en función de la cantidad indicada.
- **VELOCITY(valor):** Cambia la velocidad de la nota al valor indicado.

"Debido a esta representación, el modelo es capaz de generar composiciones con un tiempo y una dinámica más naturales en comparación con nuestros otros modelos que utilizan una cuadrícula métrica cuantificada con un tempo fijo y no manejan la velocidad de las notas."

Existen múltiples configuraciones para este modelo. Las propuestas dentro del repositorio del modelo son las siguientes:

- **Performance:** *"Ignora las velocidades de las notas pero modela los eventos de comienzo y fin con eventos de tiempo."*
- **Performance with dynamics:** *"Incluye cambios de velocidad cuantizados en 32 contenedores."*
- **Performance with dynamics and modulo encoding:** *"Utiliza una codificación alternativa diseñada por Vida Vakilotojar [44] donde los valores de los eventos son mapeados a puntos en el círculo de la unidad."*
- **Conditioned performance with dynamics:** Existen múltiples configuraciones condicionadas, aquellas condicionadas por la clase de tonos deseada, las condicionadas por la densidad de notas que tendrá ésta y una última que combina ambas.

En [45] se explica este modelo y varios aspectos del mismo así como decisiones tomadas a lo largo del desarrollo de éste. Se incluyen ejemplos que varían en función de los parámetros introducidos al modelo.

Todos estos modelos cuentan con multitud de parámetros configurables en cada una de sus configuraciones, tanto para el entrenamiento como para la generación o la creación del dataset. Debido al alcance de éste proyecto, en el capítulo 4 explicaremos aquellos que hemos utilizado o modificado en las pruebas. Dentro de los repositorios de estos modelos se encuentra una explicación detallada acerca los mismos.

3.2. MuseGAN

El proyecto **MuseGAN** nace a partir de la idea de generar composiciones musicales complejas en las que existan polifonía y varios instrumentos de forma simultánea a partir de GAN. Los responsables del proyecto ya plantearon una idea similar anteriormente con su proyecto **MidiNet** [46].

En los artículos [47] y [48] se determinan con detalle las bases de funcionamiento del modelo que sirvieron para el desarrollo del proyecto MuseGAN a partir de 2017.

Del mismo modo que Magenta, los datos se crean a partir de un conjunto de MIDIs. Sin embargo, MuseGAN dispone de su propia representación de datos ***pianoroll***. Esta representación nos permite interpretar los MIDIs como una matriz binaria en la que las columnas representan la presencia/ausencia de una nota y las filas el tono de ésta. Podemos ver un ejemplo de ésto en la figura 3.3.

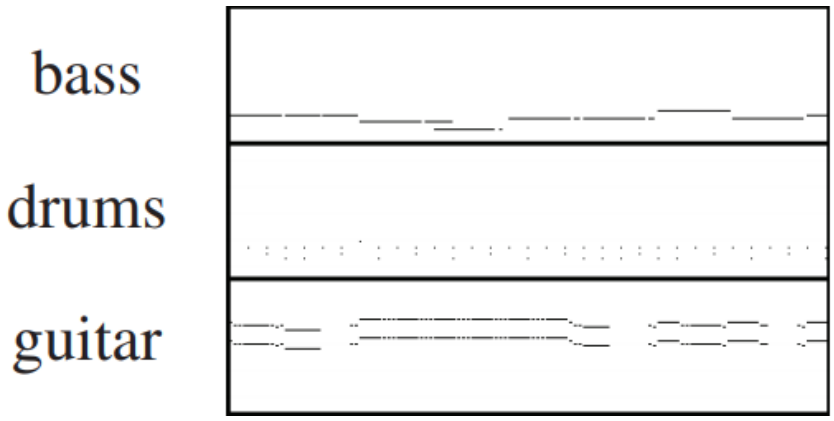


Figura 3.3: Representación MIDI en formato pianoroll. Extraída de [48]

Esta representación es la que encontramos en cualquier estación de trabajo de audio digital o **DAW** (Digital Audio Workstation) por sus siglas en inglés, a la hora de escribir melodías mediante un *piano roll*. Ejemplos de ésto los encontramos en DAWs como FL Studio o Ableton live. (figura 3.4)

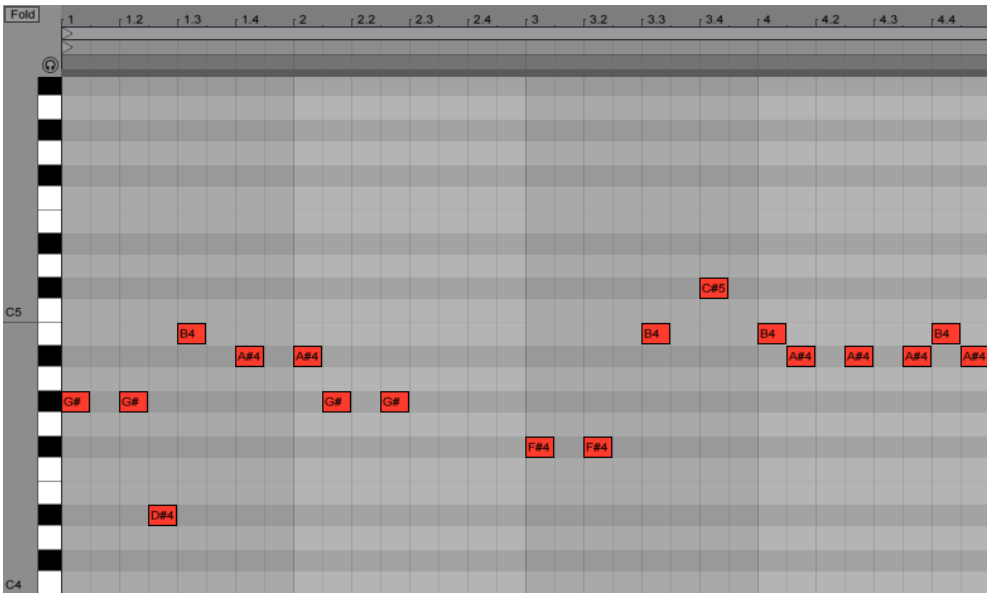


Figura 3.4: Melodía escrita mediante una herramienta piano roll en Ableton Live. [1]

[1] Extraído de: http://fadil.adtddns.asia/ableton-templates.html

Los autores comentan respecto a la composición musical: *"En nuestra experiencia, hay dos formas comunes de crear música. Dado que un grupo de músicos tocan diferentes instrumentos, pueden crear música improvisando música sin un arreglo predefinido, también conocido como **jamming**. O, podemos tener un **compositor** que arregla instrumentos con conocimiento de estructura armónica e instrumentación. Los músicos entonces tocarán según ésta."*.

A partir de este razonamiento surgen las distintas configuraciones disponibles:

- **Jamming model:** Existe un par generador-discriminador para cada instrumento. Cada uno de éstos aprende independientemente del aprendizaje del resto de pares. De esta forma se aplica el concepto de varios músicos improvisando de forma simultánea.
- **Composer model:** Existe un solo par generador-discriminador que actuaría como "compositor", creando éste las pistas para todos los instrumentos simultáneamente.
- **Hybrid model:** Misma estructura que en el modelo *jamming*, sin embargo la entrada de todos los generadores está alterada por un vector de entrada común y aleatorio, que actuaría como un "compositor" coordinando a los músicos.

Respecto a la generación existen dos configuraciones distintas. La primera nos permite **generar composiciones desde cero**, es decir, partiendo de la información aprendida en el entrenamiento. La segunda, nos permite crear **generaciones influenciadas por una pista** introducida por un humano. Estas generaciones tenderán a imitar la pista original a partir de los datos del entrenamiento y los de la pista.

Las generaciones obtenidas se almacenan en formato *pianoroll*. Como ya hemos mencionado antes, este formato está compuesto por un conjunto de matrices binarias, sin embargo, para generar éstas es necesario binarizar las salidas del generador. Para ello se utilizan dos técnicas de postprocesado, **muestreo de Bernouilli** [49, Capítulo 7] o BS (Bernouilli Sampling) por su siglas en inglés y **umbralización dura** [50] o HT (Hard Thresholding) por sus siglas en inglés. Debido a las diferencias existentes entre estas técnicas, se almacenan generaciones obtenidas a través de ambos métodos.

Poco tiempo después de presentar este modelo, propusieron una ampliación en [51]. El concepto que introdujeron fue utilizar neuronas binarias en las capas de salida de los generadores para determinar la presencia/ausencia de las notas en su formato de datos. Gracias a esta neuronas, las salidas obtenidas serían directamente binarias y por lo tanto las fases de postprocesado no serían necesarias. Según los autores, la introducción de este concepto resultó en generaciones con notas más consistentes y menos fragmentadas respecto a las obtenidas a través del postprocesado.

3.3. Musicnn

Musicnn [52] [53] (pronunciado *"musician"*) es una librería implementada por Jordi Pons y Xavier Serra enfocada en técnicas MIR a través de CNN. Las técnicas que cubre esta implementación son el **etiquetado musical**, **la extracción de características** y el **traspaso de aprendizaje**.

Dispone de varios modelos pre-entrenados sobre datasets musicales relevantes en el ámbito de las técnicas MIR, éstos son: ***MagnaTagATune (MTT)*** [54] y ***Million Song Dataset (MSD)*** [55]. En lo referente al etiquetado musical y la extracción de características, ambos datasets cuentan con su propio conjunto de (50) etiquetas para describir las composiciones que se le proporcionen.

MTT: *guitar, classical, slow, techno, strings, drums, electronic, rock, fast, piano, ambient, beat, violin, vocal, synth, female, indian, opera, male, singing, vocals, no vocals, harpsichord, loud, quiet, flute, woman, male vocal, no vocal, pop, soft, sitar, solo, man, classic, choir, voice, new age, dance, male voice, female vocal, beats, harp, cello, no voice, weird, country, metal, female voice, choral.*

MSD: *rock, pop, alternative, indie, electronic, female vocalists, dance, 00s, alternative rock, jazz, beautiful, metal, chillout, male vocalists, classic rock, soul, indie rock, mellow, electronica, 80s, folk, 90s, chill, instrumental, punk, oldies, blues, hard rock, ambient, acoustic, experimental, female vocalist, guitar, hip-hop, 70s, party, country, easy listening, sexy, catchy, funk, electro, heavy metal, progressive rock, 60s, rnb, indie pop, sad, house, happy.*

Además de los modelos pre-entrenados, podemos entrenar nuestros propios modelos con un dataset de nuestra elección a través de la herramienta **Musicnn-training** (https://github.com/jordipons/musicnn-training).

3.4. LSTM GClassifier

LSTM GClassifier [56] es una librería desarrollada por Ruoho Ruotsi y Nazar Ponochevnyi enfocada a la clasificación por géneros musicales mediante la utilización de RNN con neuronas LSTM.

Dispone de un modelo pre-entrenado sobre uno de los datasets más antiguos y extendidos en el ámbito MIR. ***GTZAN dataset*** [57] nos permite utilizar esta herramienta de forma inmediata. Las características a partir de las cuales se clasifican las composiciones introducidas son las siguientes:

- **MFCC**
- **Chroma**
- **Centroide espectral**
- **Contraste espectral**

Las dos primeras características se han mencionado en apartado 2.2. El centroide espectral [58] está relacionado con el “brillo” de un sonido, que en el ámbito musical se refiere al timbre de éste, razón por la cual supone una buena característica a analizar. Por su parte, el contraste espectral [59] representa las características espectrales relativas, aproximando así la distribución de componentes armónicos y no armónicos. Dentro de los artículos citados se encuentra mucha más información sobre estas características y sus casos de uso.

El dataset utilizado por esta herramienta nos permite separar las composiciones musicales en diez géneros: *blues, classical, country, disco, hiphop, jazz, metal, pop, reggae y y rock*.

4

Fase experimental

En este capítulo introduciremos el experimento a llevar a cabo, así como las fases que este comprende. Cada fase será descrita a fondo, indicando las dificultades encontradas, las configuraciones de cada herramienta y los scripts desarrollados con el fin de facilitar las tareas a llevar a cabo.

El experimento se divide en 3 fases. Cada una se encarga de llevar a cabo un objetivo de los mencionados en el apartado 1.2. La primera de ellas (**O-1**) comprende la obtención de datos de entrenamiento a partir de un conjunto de MIDIs. La segunda fase (**O-2**) comprende los procesos de entrenamiento de las NN usadas y la obtención de composiciones artificiales a través de éstas. Por último, la tercera fase (**O-4** e introduce **O-3**) explica cómo se aplican las técnicas MIR sobre los datos de entrenamiento y los datos obtenidos en las generaciones para posteriormente ser comparados. En la figura 4.1 podemos ver la estructura del experimento.

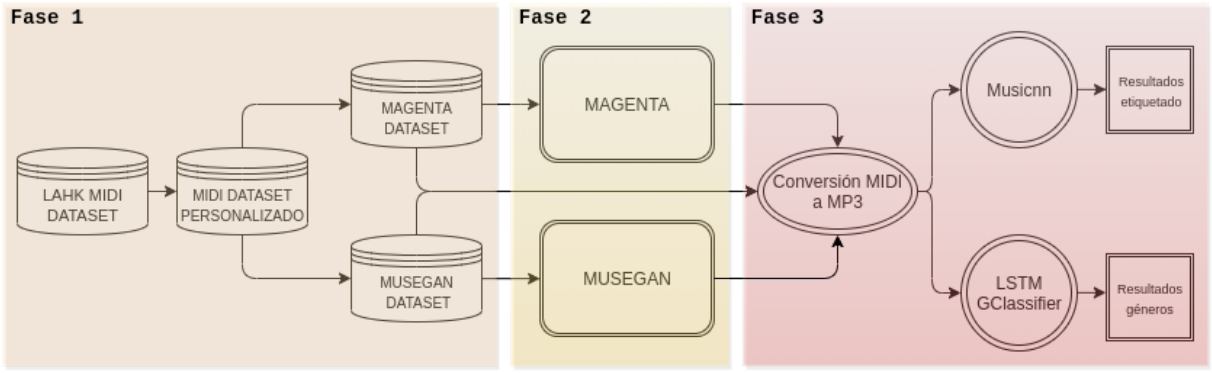


Figura 4.1: Estructura general del experimento.

4.1. Fase 1: Creación del conjunto de datos personalizado

Tal y como se indica en el **O-1**, necesitamos un conjunto de datos que nos permita trabajar con las herramientas de generación y las técnicas MIR. De esta necesidad surge el primer problema: ¿cómo podemos obtener una fuente de datos, en este caso MIDIs, ya procesada y preparada?

Una de las primeras analizadas fue el **MAESTRO DATASET**, un conjunto de datos propuesto en [60] y explicado también en uno de los artículos del blog de Magenta, que incluye datos procesados y listos para trabajar. El problema que planteaban estos datos era que solamente incluían composiciones de pianistas de carácter clásico, por lo que, de cara a la extracción de características y géneros posiblemente obtendríamos poca información. A pesar de haberlo descartado, el modelo **Performance RNN** (descrito en el apartado 3.1) aprovecharía a la perfección el detalle con el que están almacenadas estas composiciones gracias a su estructura de datos por eventos.

Uno de los requisitos que debía cumplir el conjunto de datos a usar era que fuera capaz de adaptarse a todos los modelos que íbamos a usar, tanto para Magenta como para MuseGAN. Además de ésto, debería estar compuesto por géneros musicales que se contemplasen dentro de los experimentos que íbamos a realizar en cuanto a recuperación de información. Que el conjunto de datos sea adaptable para Magenta y MuseGAN supone varios requisitos que podemos resumir en:

- MIDIs que contengan pistas monofónicas.
- MIDIs que contengan pistas polifónicas.
- MIDIs que contengan varias pistas de instrumentos (Al menos los contemplados en MuseGAN).

De cara a las herramientas MIR a utilizar, Musicnn y LSTM GClassifier, el conjunto de datos no necesita cumplir tantos requisitos ya que posiblemente cualquier conjunto que cumpla las restricciones para los modelos de generación proporcione información interesante, sin embargo, si queremos información variada para poder comparar entre las composiciones originales y las generadas es interesante buscar un conjunto similar a los usados en los modelos pre-entrenados de éstas herramientas.

Encontramos un nuevo conjunto de datos gracias al artículo escrito por Justin Shenk [61] el cual trata sobre generación de composiciones en base a un género musical mediante la herramienta Magenta. El conjunto de datos encontrado es el **Lakh MIDI Dataset** [62] [63] y es descrito por su autor, Colin Raffel, como *“una colección de 176.581 ficheros MIDI únicos, 45.129 de los cuales han sido emparejados y alineados con las entradas del Million Song Dataset. Su objetivo es facilitar la recuperación de información musical a gran escala, tanto simbólica (utilizando sólo los archivos MIDI) como basada en el contenido de audio (utilizando la información extraída de los archivos MIDI como anotaciones para los archivos de audio emparejados).”*.

Además del conjunto de MIDIs completo, existen múltiples subconjuntos de éste con distintas características. Podemos encontrar estos en [63]. Aquel que mejor se ajusta a nuestra necesidad es el ***Clean MIDI subset***, ya que es el único en el que los ficheros MIDIs están separados por artistas y nombrados por el título de la canción que representan, por lo que, a pesar de ser un subconjunto parece ser más útil que el resto de cara a separar por géneros los MIDIs. Examinando los ficheros que componen éste observamos que la mayoría de los ficheros contienen pistas monofónicas y polifónicas así como varios instrumentos, por lo que cumple también con los requisitos para los modelos de ge-

neración. Por último, tal y como se menciona en la descripción de este conjunto de datos, muchos de los MIDIs que conforman éste forman parte del Million Song Dataset, mencionado en el apartado 3.3. Debido a que cumple todos los requisitos expuestos anteriormente decidimos utilizar este conjunto de datos en su modalidad ***Clean MIDI subset***.

Una vez seleccionado el conjunto de datos, nuestra siguiente tarea es la de determinar qué géneros formarán parte del conjunto de datos final. A pesar de que el tamaño de este subconjunto sea de 17257 ficheros, es decir, un 10 % del total, debido a la carga computacional que suponen los modelos de generación en sus fases de conversión de datos y de entrenamiento debemos eliminar los ficheros que menos información aporten e intentar comprimir estos en un subconjunto de *música pop*. Con *música pop* nos referimos a aquellos géneros que más tendencia han marcado en las últimas décadas dentro de la industria de la música en términos comerciales. Ejemplos de esto son: *pop, rock, rnb, soul, dance*, etc.

Para decidir qué géneros formarán parte del conjunto de datos final necesitamos ser capaces de automatizar la separación por géneros a partir de la clasificación que nos ofrece el ***Clean MIDI subset***. Como hemos mencionado antes, en [61] se trata la separación por géneros de este dataset, por lo que siguiendo los pasos mostrados en el artículo creamos un script en python que nos permita hacer esto. Uno de los requisitos para poder llevar a cabo esta tarea es utilizar la API de Spotify [64]. Gracias a ésta podemos realizar una serie de peticiones relacionadas con canciones, usuarios, playlists, etc. Mediante la librería ***Spotipy*** [65] podemos hacer uso de esta API de una forma sencilla e intuitiva gracias a su extensa documentación.

De entre las funciones disponibles, aquella que mejor se ajusta a nuestras necesidades es la función *search* ya que nos permite obtener información sobre cualquier tipo de ítem que exista en Spotify. Gracias a que disponemos de los nombres de los artistas así como de las canciones, podemos obtener información a partir de cualquiera de éstos, sin embargo, con el objetivo de agilizar éste proceso realizamos la búsqueda directamente por artistas. Dentro de la información devuelta por esta función se encuentran los géneros más comunes en las composiciones de estos artistas. Una vez tenemos los géneros de cada artista los almacenamos a modo de diccionario en un fichero **json**.

{ “The Outhere Brothers”: [“eurodance”, “hip house”], “Friend & Lover”: [“bubblegum pop"], “Bruce Channel”: [“rock-and-roll"], “Loverboy”: [“album rock”, “classic canadian rock”, “classic rock”, “glam metal”, “hard rock”, “heartland rock”, “mellow gold”, “new wave pop”, “rock”, “soft rock”], ... }

Cuadro 4.1: Géneros musicales recuperados por Artista mediante Spotify API y almacenados en fichero JSON.

Revisando los géneros obtenidos, observamos que existen multitud de subgéneros que pueden

complicar la tarea de separación por géneros, tal y como se ve en el cuadro de texto 4.1. Es por esta razón que se toma la decisión de incluir todos los subgéneros de un género en particular a partir de una sola búsqueda. Ésto se vería reflejado en nuestro fichero *json* de tal forma que si buscamos por el género *rock* se devuelve cualquier artista que contenga en alguno de sus géneros la palabra rock. A partir de esta idea se crea un nuevo modo de ejecución para el script que nos permite introducir los géneros que queremos recuperar. Estos géneros son recuperados desde el fichero *json* que le indiquemos al script por parámetro. Mediante una búsqueda por valores en el fichero recuperamos todos los artistas que corresponden a los géneros indicados. A partir de ésto se generan datasets diferenciados por cada género. De cara a la selección de géneros que formarán el conjunto final de datos, se seleccionan aquellos que mayor presencia tienen y que más géneros engloban. Éstos son:

- Pop
- Rock
- Rnb
- Soul
- Synth
- Wave

Géneros como el *reggae*, el *hip-hop* o el *blues* devuelven pocos resultados y encuentran representación en algunos de los géneros que sí se recuperan, como por ejemplo el *soul*, el *rnb* o el *blues*. El género *clásico* prácticamente no tiene presencia dentro del conjunto de datos original por lo que también es descartado. Varios de los artistas recuperados comparten géneros y por lo tanto estarán repetidos en los datasets generados, sin embargo, al juntar todos los artistas dentro de un mismo dataset eliminamos todas estas repeticiones. Finalmente, con todos los ficheros obtenidos por géneros contamos con una colección de 14109 MIDIs que comprenderán nuestro **conjunto de datos personalizado**. El script utilizado para llevar a cabo la separación por géneros musicales se encuentra en el apéndice A.

El siguiente paso es adaptar nuestro conjunto de datos a los modelos de Magenta y MuseGAN que utilizaremos en la generación de composiciones. Cada herramienta requiere un formato de datos distinto para entrenar, por lo que describiremos cada uno por separado. Como ya mencionamos en el apartado 3.1, utilizamos el formato *NoteSequences* en los modelos de Magenta. Convertimos nuestros datos mediante la API usando el siguiente comando:

```
convert_dir_to_note_sequences  –input_dir=$INPUT_DIRECTORY
–output_file=$SEQUENCES_TFRECORD –recursive
```

Cuadro 4.2: Comando para transformar MIDIs a formato NoteSequences.

Dónde el parámetro **input_dir** toma el valor del directorio en el que tenemos almacenado nuestro

conjunto de MIDIs y el parámetro **output_file** recibe el nombre del fichero dónde almacenaremos los datos en formato *NoteSequences* (debe tener extensión ".tfrecord"). El parámetro **recursive** indica que en caso de existir carpetas hijas también se tengan en cuenta en la conversión. Una vez ejecutado este comando obtenemos el fichero con los datos convertidos, al cual nos referiremos a partir de ahora como **data.tfrecord**.

Cada uno de los modelos utilizados procesa los MIDIs según su funcionamiento por lo que es necesario realizar tres conversiones a partir del fichero **data.tfrecord**.

Melody RNN se encarga de extraer melodías de los MIDIs que se le han proporcionado. Durante este proceso de transformación, se pueden descartar MIDIs por varias razones como por ejemplo, un rango de tonos muy amplio, MIDIs corruptos, melodías muy cortas, muy largas, etc.

```
melody_rnn_create_dataset  –config=$CONFIG –input=data.tfrecord
–output_dir=$OUTPUT_DIR –eval_ratio=0.1
```

Cuadro 4.3: Comando para convertir los datos al modelo Melody RNN.

Polyphony RNN al tratarse de un proceso más complejo en el que se contemplan melodías polifónicas requiere un procesado de los datos más selectivo. Del mismo modo que el modelo anterior se descarta toda la información que esté corrupta, incompleta o no sirva para el modelo.

```
polyphony_rnn_create_dataset  –input=data.tfrecord
–output_dir=$OUTPUT_DIR –eval_ratio=0.1
```

Cuadro 4.4: Comando para convertir los datos al modelo Polyphony RNN.

Performance RNN al contemplar eventos en los datos procesados requiere de un procesado aún más complejo que los anteriores modelos, llegando a generar un fichero de datos de entrenamiento de 121 GB y con un tiempo de ejecución de alrededor de 6-7h. Del mismo modo que los modelos anteriores también descarta información "defectuosa".

```
performance_rnn_create_dataset  –config=$CONFIG
–input=data.tfrecord –output_dir=$OUTPUT_DIR
–eval_ratio=0.10
```

Cuadro 4.5: Comando para convertir los datos al modelo Performance RNN.

Todos estos modelos tienen dos parámetros en común: **output_dir** almacena el valor del directorio

donde se guardarán los datos de entrenamiento y de evaluación y **eval_ratio** que indica la proporción de datos destinados al conjunto de datos de evaluación. Los modelos Melody RNN y Performance RNN añaden el parámetro **config** que permite indicar la configuración elegida (Explicadas en el apartado 3.1).

Para el modelo de MuseGAN el procedimiento es más complejo ya que no dispone de un comando que directamente nos transforme los datos, por lo que tenemos que recurrir a otras herramientas para transformar éstos. Estas herramientas se mencionan en el repositorio de MuseGAN por lo que son facilmente accesibles, sin embargo la información acerca de como transformar los datos es inexacta y requiere de una lectura en profundidad de los scripts que llevan a cabo esta operación.

La primera herramienta que utilizamos se llama ***lakh-pianoroll-dataset*** [66]. Gracias a esta podemos convertir nuestra colección de MIDIs al formato *pianoroll* multipista. Dentro de la carpeta *src* encontramos el script *converter.py* que nos permite realizar esta conversión mediante el siguiente comando:

```
python converter.py $INPUT_DIRECTORY $OUTPUT_DIRECTORY –midi-info-path
$OUTPUT_FILE_NAME
```

Cuadro 4.6: Comando para convertir los datos al formato pianoroll multipista.

Dónde el primer parámetro indica el directorio en el que se encuentran nuestros MIDIs y el segundo el directorio en el queremos almacenar las conversiones. El último parámetro es obligatorio e indica el nombre de un fichero en formato *json* donde se escribirán metadatos de cada fichero MIDI.

Con los datos en formato *pianoroll* pasamos a utilizar la herramienta ***5-track-pianoroll*** [67]. El objetivo de ésta es transformar los datos a vectores de tensorflow, también llamados *tensores* y almacenarlos en un solo fichero con formato ***numpy***. El proceso simplemente requiere realizar dos llamadas, primero, al script *parser.py* y después a *compile.py*. Para indicar de dónde deben obtener los datos modificamos ciertas variables internas de éstos. Finalmente, obtenemos un fichero con formato *numpy* que podremos introducir a nuestro modelo como datos de entrenamiento.

Con los datos de entrenamiento preparados para ambos modelos damos por terminada la primera fase del experimento. Ampliando la fase 1 de la figura 4.1 podemos ver en detalle el proceso llevado a cabo.

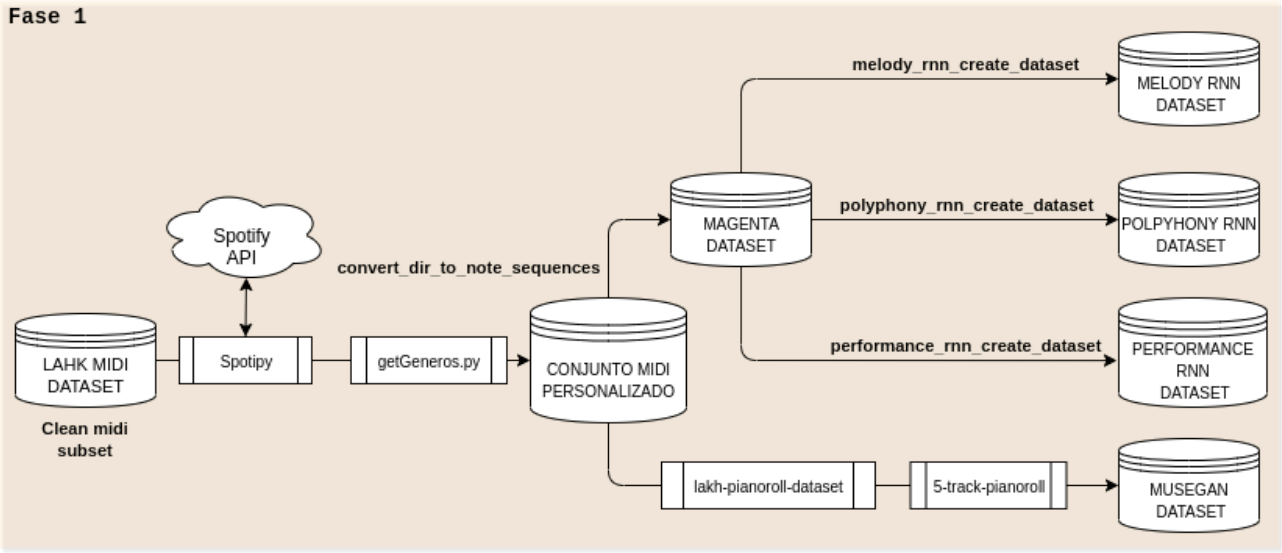


Figura 4.2: Fase 1 del experimento en detalle.

4.2. Fase 2: Entrenamiento y generación de composiciones

A fin de justificar los valores obtenidos en distintas fases de esta sección se indican a continuación las características del equipo donde se han realizado estas pruebas:

- **GPU:** NVIDIA RTX 2070 SUPER 8 GB GDDR6 (Fabricante MSI)
- **CPU:** AMD Ryzen 5 3600 6-Core, 12-Thread, 4.2Ghz
- **RAM:** Corsair Vengeance LPX Black 16GB 3000Mhz DDR4

La gran mayoría de los procesos indicados en las secciones posteriores se han llevado a cabo enteramente en la GPU, sin embargo, ha sido necesario introducir en todos los modelos usados el siguiente fragmento de código para evitar que éstos solicitasen más memoria de la disponible.

Código 4.1: Código necesario para que los modelos no soliciten VRAM a la GPU cuando ésta no dispone de más.

```
1 import tensorflow.compat.v1 as tf
2
3 gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.8)
4 config = tf.ConfigProto(gpu_options=gpu_options)
5 config.gpu_options.allow_growth = True
6 session = tf.Session(config=config)
```

Gracias a ésto limitamos la memoria de la GPU al 80 % (Valores superiores desbordan también la memoria en ocasiones) e indicamos a tensorflow que trabaje con la memoria indicada. A pesar de resultar en entrenamientos más lentos, es completamente necesario con el equipo del que disponemos.

4.2.1. Entrenamiento Magenta

Configuración del entrenamiento: parámetros e hiperparámetros

Como ya hemos visto en la sección anterior, cada modelo de Magenta tiene sus propios comandos a la hora de llevar a cabo acciones como crear el conjunto de datos, entrenar o generar. De cara al entrenamiento, la mayoría de los parámetros son comunes, existiendo algunos propios. Los parámetros comunes son:

- **run_dir:** *"Ruta al directorio donde se guardarán los puntos de control y los resúmenes de los eventos durante el entrenamiento y la evaluación. Se crearán subdirectorios separados para los eventos de entrenamiento y los eventos de evaluación dentro del directorio run_dir."* Por defecto '/tmp/model_rnn/logdir/run1'.
- **sequence_example_file:** *"Ruta al archivo TFRecord que contiene los registros para el entrenamiento o la evaluación. También se puede proporcionar un patrón de archivo, que se ampliará a todos los archivos que coincidan."* Sin valor por defecto.
- **num_training_steps:** *"El número de pasos de entrenamiento que su modelo debe dar antes de terminar. Dejar a 0 para terminar de forma manual."* Por defecto 0.
- **num_checkpoints:** *"El número de puntos de control más recientes para mantener en el directorio de entrenamiento. Guarda todo si es 0."* Por defecto 10.
- **eval:** *"Si es **True**, este proceso sólo evalúa el modelo y no actualiza los pesos."* Por defecto **False**.
- **num_eval_examples:** Solo aplica si el parámetro **eval** tiene el valor **True**. *"El número de ejemplos de evaluación que su modelo debe procesar para cada paso de la evaluación. Dejar a 0 para utilizar todo el conjunto de evaluación."* Por defecto 0.
- **summary_frequency:** *"Se registrará un resumen del proceso cada **summary_frequency** pasos durante el entrenamiento o la evaluación."* Por defecto 10.
- **log:** *"El umbral de qué mensajes se registrarán: DEBUG, INFO, WARN, ERROR, o FATAL."* Por defecto INFO.
- **hparams:** *"Lista de pares "nombre=valor" separados por comas. Para cada par, el valor del hiperparámetro llamado "nombre" se establece como "valor". Este mapeo se combina con los hiperparámetros por defecto."* Los hiperparámetros por defecto son dependientes de cada modelo.
- **config:** Exclusivo de Melody RNN y Performance RNN.*"La configuración a utilizar"*. Sin valor por defecto, es necesario indicar una configuración.

El modelo Performance RNN cuenta con el parámetro **warm_start_bundle_file** que nos permite iniciar los pesos de entrenamiento a los almacenados en un fichero de tipo bundle (.mag).

Los hiperparámetros vienen definidos por defecto en cada modelo, pero podemos modificarlos para ajustarlos a nuestras preferencias. Dentro de las configuraciones que vamos a usar, los hiperparámetros comunes definidos los encontramos en la tabla 4.1. *batch_size* hace referencia al número de ejemplos que el modelo recibe en cada iteración, *rnn_layer_sizes* representa a través de una lista el número de neuronas de cada capas que utilizará la red, *dropout_keep_prob* es la probabilidad de mantener la salida de cualquier neurona del módelo, *clip_norm* indica el máximo valor aceptado en el proceso de *clipping* (Técnica para regular los valores atípicos o *outliers*) y por último, *learning_rate* que indica la tasa de aprendizaje.

Conf.	batch_size	rnn_layer_sizes	dropout_keep_prob	clip_norm	learning_rate
Lookback	128	[128, 128]	0.5	5	0.001
Attention	128	[128, 128]	0.5	3	0.001
Polyphony	64	[256, 256, 256]	0.5	5	0.001
Performance	64	[512, 512, 512]	1.0	3	0.001

Tabla 4.1: Tabla con los hiperparámetros por defecto de los modelos usados.

La configuración **attention** contiene un hiperparámetro propio relativo a su característica principal mencionada en el apartado 3.1, la atención. Éste es **attn_length** y representa el tamaño del vector de atención, por lo que su valor está directamente relacionado con el proceso de almacenamiento en memoria de las neuronas a la hora de entrenar. Un valor mayor implicaría contemplar más información pasada dentro de la composición a la hora de introducir nueva información dentro de la neurona LSTM.

El modelo Performance RNN con su configuración *multiconditioned_performance_with_dynamics* tiene una serie de hiperparámetros propios. El primero, **num_velocity_bins** indica el número de contenedores de velocidad. Por defecto tiene el valor 32 y en caso de asignarle el valor 0, eliminamos la capacidad de leer eventos de velocidad. El segundo, **control_signals** tiene como valor una lista de objetos propios de los eventos usados en los modelos condicionados. Si no existe condicionamiento, se debe indicar el valor **None**.

Una vez comprendidos todos los parámetros de cada modelo pasamos a los comandos ejecutados para entrenar cada una de las configuraciones elegidas. Para el modelo Melody RNN se eligen las configuraciones **Lookback** y **Attention**.

```
melody_rnn_train –config=$CONFIG –run_dir=$RUN_DIRECTORY
–sequence_example_file=$CONFIG_TRAINING_TFRECORD
–num_training_steps=20000
```

Cuadro 4.7: Comando para entrenar las configuraciones lookback y attention (Melody RNN).

En el cuadro de texto 4.7 se ha generalizado el comando para ambas configuraciones ya que las únicas diferencias residen en el valor del parámetro *sequence_example_file*. El resto de valores son los indicados en las configuraciones por defecto.

Para el modelo Polyphony RNN y el modelo Performance RNN en su modalidad *multiconditioned_performance_with_dynamics* se han ejecutado los siguientes comandos:

```
polyphony_rnn_train  –run_dir=$RUN_DIRECTORY –num_training_steps=20000
–sequence_example_file=$CONFIG_TRAINING_TFRECORD
–hparams="batch_size=64, rnn_layer_sizes=[128,128,128]"
```

Cuadro 4.8: Comando para entrenar el modelo Polyphony RNN.

```
performance_rnn_train  –config=multiconditioned_performance_with_dynamics
–sequence_example_file=$CONFIG_TRAINING_TFRECORD
–run_dir=$RUN_DIRECTORY
```

Cuadro 4.9: Comando para entrenar el modelo Performance RNN en su configuración *multiconditioned_performance_with_dynamics*.

Información sobre el entrenamiento

Una vez los entrenamientos se han completado podemos observar información sobre éstos a partir de la herramienta **tensorboard** y el conjunto de métricas del que disponen estos modelos. A pesar de entrenar más las redes a partir de este punto la mejora no es significativa. Estas métricas representan:

- **Accuracy:** Mide la similitud de las entradas con las salidas obtenidas. Al tratarse de la fase de entrenamiento y de una RNN ésta no representa si un ejemplo está bien clasificado o no, como ocurre en otras situaciones.
- **Event accuracy:** Igual que *Accuracy* pero en vez de medir sobre toda la información, mide sobre los instantes en los que hay información sobre notas.
- **No event_accuracy:** Igual que *Event accuracy* pero para los casos en los que no hay información sobre notas.
- **Loss:** Representa la pérdida de la red en cada iteración. Esta se obtiene a través de la función que mide el error.
- **Perplexity:** Métrica utiliza como medida del error de predicción. A mayor valor implica mayor error en el modelo.
- **Global_step:** Indica el número de pasos realizados por segundo. De ésta forma podemos calcular el tiempo estimado de entrenamiento de cada modelo. Por ejemplo, para el modelo Polyphony, si damos un 0.1685 pasos cada segundo, para dar 20000 pasos nuestro modelo tardará alrededor de 33h en entrenar.

Podemos ver los valores finales y los mejores resultados de cada métrica en la siguiente tabla.

4.2.2. Generaciones Magenta

De cara a obtener las generaciones, Magenta nos proporciona una serie de comandos muy similares a los utilizados en el entrenamiento. Éstas representan las composiciones, en formato MIDI, que nuestros modelos han aprendido a generar, lo cual facilita mucho la interpretación de los resultados y

Conf.	Accuracy	Event accuracy	Loss	No event_acc	Perplexity	Global_step
Lookback	0.7668	0.7664	0.8072	0.6993	2.246	**2.268**
Attention	**0.7981**	**0.7995**	**0.7176**	0.7617	**2.054**	1.287
Polyphony	0.7013	0.6526	1.106	0.9292	3.034	0.1685
Performance	0.67	0.6653	1.09	**0.937**	3.283	1.925

Tabla 4.2: Métricas de entrenamiento en Magenta de los modelos usados.

nos permite trabajar con éstos en multitud de aplicaciones. Cada modelo cuenta con un comando propio para obtenerlas y algunos parámetros exclusivos. Primero introduciremos los comandos de forma general y después explicaremos los parámetros utilizados en las distintas generaciones obtenidas.

```
melody_rnn_generate –config=$CONFIG –run_dir=$RUN_DIRECTORY
–output_dir=$OUTPUT_DIRECTORY –num_outputs=$NUM_OUTPUTS
–num_steps=$NUM_STEPS –primer_melody=$EXAMPLE_MELODY
```

Cuadro 4.10: Comando para generar melodías en las configuraciones lookback y attention (Melody RNN).

```
polyphony_rnn_generate –run_dir=$RUN_DIRECTORY
–hparams="batch_size=64, rnn_layer_sizes=[128,128,128]"
–output_dir=$OUTPUT_DIRECTORY –num_outputs=$NUM_OUTPUTS
–num_steps=$NUM_STEPS –condition_on_primer=$BOOLEAN
–inject_primer_during_generation=$BOOLEAN
–primer_pitches=$EXAMPLE_CHORD
```

Cuadro 4.11: Comando para generar melodías en el modelo Polyphony RNN.

Para que estos comandos funcionen correctamente, los parámetros comunes al entrenamiento deben ser exactamente los mismos. Esto afecta a los parámetros *run_dir*, *hparams* en caso de haberse modificado y *config* en caso de existir múltiples configuraciones. El resto de parámetros en su gran mayoría son comunes a todos los modelos por lo que los listaremos a continuación:

- **output_dir:** Directorio dónde se almacenarán las melodías generadas.
- **num_outputs:** Número de melodías a generar. Por defecto 10.
- **num_steps:** Longitud de las melodías generadas. 16 pasos equivalen a un compás en 4/4. Por defecto 128 (8 compases).
- **qpm:** Tempo de las melodías generadas. Por defecto 120 *beats per minute*.

```
performance_rnn_generate  –config=$CONFIG –run_dir=$RUN_DIRECTORY
–output_dir=$OUTPUT_DIRECTORY –num_outputs=$NUM_OUTPUTS
–num_steps=$NUM_STEPS –primer_melody=$EXAMPLE_MELODY
```

Cuadro 4.12: Comando para generar melodías en el modelo Performance RNN en su configuración *multiconditioned_performance_with_dynamics*.

- **temperature:** Aleatoridad a la hora de seleccionar las notas de las melodías. A un mayor valor existirá una mayor aleatoridad. Por defecto 1.
- **primer_melody:** *Lista Python de eventos NoteSequence (-2 = Sin evento, -1 = Evento fin de nota, velores entre 0 y 127 = Eventos comienzo de nota en escala de tonos MIDI) en formato textual. Por ejemplo: "[60, -2, 60, -2, 67, -2, 67, -2]".* La melodía de ejemplo tiene las notas do (60) y sol (67) en una de sus escalas. Los '-2' representan silencios. Cada incremento en los valores de las notas equivale a subir un semitono, por lo que a mayores valores tendremos tonos más agudos.
- **primer_midi:** *"La ruta a un archivo MIDI que contiene una pista monofónica/polifónica que se usará como inspiración para la generación."*
- **primer_pitches:** Exclusivo de los modelos polifónicos. *"Lista Python de tonos que será usada como un acorde inicial de corta duración. Por ejemplo: "[60, 64, 67]""* Ayuda a marcar la tonalidad de la melodía. El ejemplo es un acorde en do mayor.

En el modelo Polyphony RNN existen un par de parámetros exclusivos. El primero, **condition_on_primer**, según su descripción, *"Si está activado, la RNN recibirá el **primer** como su entrada antes de que empiece a generar una nueva melodía. Lo más probable es que quieras que esté activado si estás usando "primer_pitches" para iniciar la melodía con un acorde y establecer una cierta tonalidad. Si estás usando "primer_melody" porque quieres inyectar una melodía en la salida usando "inject_primer_during_generation", es probable que quieras que no esté activado, de lo contrario el modelo verá una melodía monofónica antes de que se le pida que produzca una secuencia polifónica."* Por su parte, **inject_primer_during_generation**, *"si se activa, el **primer** se inyectará como parte de la secuencia generada. Esta opción es útil si se quiere que el modelo armonice una melodía existente. La configuración óptima para usar esta opción es con "primer_melody" y "condition_on_primer" desactivado"*.

El modelo Performance RNN añade un par de parámetros a la generación en el caso de utilizar uno de los modelos condicionados. **notes_per_second** permite incrementar el número de notas por segundo en la melodía generada. **pitch_class_histogram** representa la presencia y frecuencia de las notas que sonarán en la melodía a partir de una lista python en formato string. Por ejemplo: "[2, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]". Esta lista representa la escala do mayor. Con un 0 se indican los tonos que no estarán presentes en la melodía y con valores mayores a 0 la frecuencia con la que aparecerán. Por ejemplo, la nota do aparecerá el doble de veces que el resto de notas.

Los parámetros configurados en las generaciones obtenidas parten de las configuraciones más

usadas en las composiciones musicales dentro de la “música pop”. El ***qpm*** que se ha configurado para todas las melodías es ’110.0’ *beats per minute*. A pesar de haber fijado este valor, esto no supone ningún problema ya que cualquier aplicación en la que utilicemos el MIDI generado nos permitirá modificar su tempo. Respecto a la longitud de las melodías, todas las melodías se han generado con 16 compases, sin tener en cuenta la longitud que puedan añadir los parámetros de tipo **primer**. Este tipo de parámetros se ha utilizado para ayudarnos a marcar las tonalidades de las melodías generadas a pesar de que el componente aleatorio de éstas no asegura que podamos fijar las generaciones en una tonalidad concreta. Para el parámetro *primer_melody* se han utilizado los siguientes valores:

- Melodía en Do mayor: “[60,-2,65,71,69,-2,74,-2,76,-2,-2,-2,67,-2,62,-2]”
- Melodía en Mi menor: “[64,-2,66,-2,74,-2,69,-2]”
- Melodía en Fa sostenido menor: “[66,-2,69,-2,73,-2,69,64]”
- Melodía en Sol insen: “[77,-2,68,-2,55,-2,-2,-2,72,68,65,56,72,-2,55,-2]”

Las dos primeras escalas seleccionadas son muy comunes en la música mientras que las otras dos se han seleccionado para ver como el modelo actúa frente a datos menos frecuentes. Este parámetro se ha aplicado a las generaciones de los modelos Melody RNN y Performance RNN. Para el modelo Polyphony RNN se ha utilizado el parámetro *primer_pitches* con los siguientes acordes:

- Si menor séptima: “[59,62,66,69]”
- Sol dominante séptima con fa invertido: “[71,67,74,65]”
- Mi menor: “[64,71,67]”
- Do mayor: “[67,64,60]”
- Re menor: “[62,65,69]”

Al utilizar este parámetro, los parámetros *condition_on_primer* y *inject_primer_during_generation* estaban a **true** y **false** respectivamente. Debido a los resultados obtenidos con este modelo se han probado más configuraciones y se ha añadido el parámetro *temperature* variando su valor entre ’1’ y ’1.2’ con el fin de darle más variedad y sentido a las composiciones.

Los parámetros que no se han mencionado dentro de la configuración de los resultados, o bien no se han usado, o bien han mantenido sus valores por defecto. Existen multitud de configuraciones posibles y de parámetros que permiten optimizar tanto los procesos de entrenamiento como los de generación de cara a obtener los mejores resultados posibles, sin embargo, esto conllevaría una batería de pruebas excesiva que escapa del alcance del proyecto.

4.2.3. Entrenamiento MuseGAN

Configuración de experimentos

Uno de los mayores problemas encontrados de cara a comenzar a usar MuseGAN es la falta de documentación existente sobre esta herramienta. Al contrario que Magenta, MuseGAN no es un modelo tan extendido debido a su reciente aparición, estando, en consecuencia, mucho menos difundido. Debido a la poca información disponible sobre los parámetros de configuración, los procesos de entrenamiento y generación se han visto condicionados y no se aporta información tan detallada, como se hizo en el modelo anterior.

Como ya mencionamos al final del apartado 3.2, el último artículo presentado por los autores de este modelo propone una estructura diferente en la que se utilizan neuronas binarias en las capas de salida de los generadores. Se proponen por lo tanto, dos configuraciones diferentes para el proceso de entrenamiento, siendo la primera aquella que utiliza la estructura original de MuseGAN y la segunda aquella que utiliza neuronas binarias.

Para poder comenzar el entrenamiento primero debemos crear un *experimento*. Esto lo hacemos mediante el siguiente comando:

```
./scripts/setup_exp.sh $EXP_DIRECTORY $EXP_DESCRIPTION
```

Cuadro 4.13: Comando para crear un *experimento* en MuseGAN.

Dónde el parámetro **$EXP_DIRECTORY** indica el directorio donde queremos almacenar nuestro experimento y **$EXP_DESCRIPTION** nos permite introducir una pequeña descripción del mismo. Creamos por lo tanto un experimento para cada configuración. Nos referiremos a ellos a partir de ahora como experimento **classic** y experimento **binario**.

En los directorios indicados en el comando anterior encontramos una serie de ficheros y subdirectorios que contienen información exclusiva de estos experimentos. Las configuraciones por defecto de éstos se encuentran en los ficheros **config.yaml** y **params.yaml**. En el primero se encuentran todos los parámetros que afectan al proceso de entrenamiento, mientras que en el segundo encontramos los parámetros relativos a los datos y la estructura del modelo.

Para llevar a cabo el experimento binario debemos realizar ciertos cambios en la configuración. En el fichero *config.yaml*, es necesario cambiar el parámetro **use_slope_annealing** a **true** y en el fichero *params.yaml*, debemos cambiar el parámetro **use_binary_neurons** a **true** y el parámetro **generator** al valor **dnb**. El resto de parámetros modificados son comunes a ambos experimentos y solo afectan al fichero *config.yaml*. Éstos son:

- **save_samples_steps:** Se ha cambiado a 0 para deshabilitar esta función y acelerar el proceso de entrenamiento.
- **save_checkpoint_steps:** Se ha disminuido su valor de 10000 a 1000 para que durante el entrenamiento se

guarden más puntos de control a partir de los cuales continuar en caso de tener que terminar el proceso de entrenamiento por cualquier razón.

- **evaluate_steps:** Se ha cambiado a 0 para deshabilitar esta función y acelerar el proceso de entrenamiento.
- **data_source:** Tipo de formato del conjunto de datos de entrenamiento. Se cambia a **npy**.
- **data_filename:** Nombre del fichero del conjunto de datos. Este fichero es el obtenido al final del proceso de creación de datos de MuseGAN visto en el apartado 4.1.
- **steps:** Se aumenta su valor de 50000 a 240000 ya que los entrenamientos previos con ese número de pasos parecen ser insuficientes para obtener buenos resultados. Este parámetro no representa exactamente el número de iteraciones que realiza el modelo, por lo que se establece en un número mucho mayor a fin de detenerlo cuando este alcance 20000 iteraciones, del mismo modo que en Magenta.
- **tempo:** Se cambia de 100 a 110. Mismo valor que en Magenta.

Todas las modificaciones que implican deshabilitar funciones han sido realizadas debido a la exigencia del modelo MuseGAN en cuanto a coste computacional. A pesar de las modificaciones comentadas al comienzo de esta subsección y las realizadas en las configuraciones, las trazas durante el comienzo del entrenamiento sugerían que el modelo requería de 3,6 GB más de memoria en la GPU.

Es importante mencionar que los modelos introducidos en el apartado 3.2 no son mencionados en el repositorio como posibles configuraciones. Tampoco se ha encontrado información sobre éstos en los ficheros de configuración del modelo. Es por esto que el proceso de entrenamiento y configuración se basa en la configuración por defecto, a excepción de los parámetros mencionados anteriormente.

Con los dos *experimentos* configurados podemos lanzar el proceso de entrenamiento de estos a partir del siguiente comando:

```
./scripts/run_train.sh $EXP_DIRECTORY $GPU_DEVICE
```

Cuadro 4.14: Comando para crear un experimento en MuseGAN.

Dónde el primer parámetro hace referencia al experimento que queremos entrenar y el segundo al índice que ocupa la GPU a usar dentro del listado de GPUs del sistema. En caso de solo disponer de una, el valor a introducir es '0'.

Información sobre el entrenamiento

Consideramos que los entrenamientos están completos cuando estos alcanzan las 20000 iteraciones. Durante éstos, podemos ver a través de las trazas mostradas la evolución de la métrica **loss** para el generador y el discriminador. Como ya mencionamos en las métricas de Magenta "representa la pérdida de la red en cada iteración. Ésta se obtiene a través de la función para medir el error".

Por desgracia esta es la única métrica que muestra el modelo por lo que no podemos saber nada más acerca del entrenamiento hasta que apliquemos las técnicas MIR sobre las generaciones.

Experimento	Generador	Discriminador
Clásico	-8.78037E+01	**-1.09111E+01**
Binario	**-5.48174E+01**	-1.38411E+01

Tabla 4.3: Valores métrica 'loss' en MuseGAN tras entrenamiento.

4.2.4. Generaciones MuseGAN

A la hora de obtener generaciones, disponemos de dos alternativas, a través de **inferencia** y a través de **interpolación**. La única información de la que disponemos sobre ésto es que la segunda realiza una interpolación de la información aprendida durante el entrenamiento en el espacio latente del modelo. Entendemos por lo tanto que, el primer modelo simplemente realiza generaciones a partir del generador, sin haber aplicado ningún tipo de modificación a los datos aprendidos durante el entrenamiento.

Del mismo modo que en Magenta, existe la posiblidad de influenciar nuestras generaciones a partir de una pista de ejemplo. Para ello, es necesario modificar ciertos parámetros en el fichero *params.yaml*. No se ha contemplado esta opción ya que el objetivo es obtener resultados más generales que representen múltiples tipos de composición y no solo uno.

Podemos aplicar los dos tipos de generación a partir de los siguientes comandos:

```
./scripts/run_inference.sh $EXP_DIRECTORY $GPU_DEVICE

./scripts/run_interpolation.sh $EXP_DIRECTORY $GPU_DEVICE
```

Cuadro 4.15: Comandos para generar composiciones en el modelo MuseGAN.

Siendo los parámetros precedidos por $ los mismos que en la etapa de entrenamiento.

Las generaciones obtenidas se almacenan en formato *pianoroll*. Tal y cómo se mencionó en el apartado 3.2, *"Esta representación nos permite interpretar los MIDIs como una matriz binaria en la que las columnas representan la presencia/ausencia de una nota y las filas el tono de ésta"*. Para generar estas matrices binarias, el modelo clásico necesita realizar un postprocesado sobre los valores reales obtenidos en las salidas. Este postprocesado lo realiza mediante dos técnicas distintas, **muestreo de Bernouilli** o BS (Bernouilli Sampling) por su siglas en inglés y **umbralización dura** o HT (Hard Thresholding) por sus siglas en inglés. Debido a las diferencias entre estas técnicas, se generan composiciones para ambos métodos. Gracias al formato de éstas, podemos convertir fácilmente las composiciones a formato MIDI, lo que nos facilitará la aplicación de técnicas MIR.

Ampliando la fase 2 de la figura 4.1 podemos ver en detalle el proceso llevado a cabo en la figura 4.3.

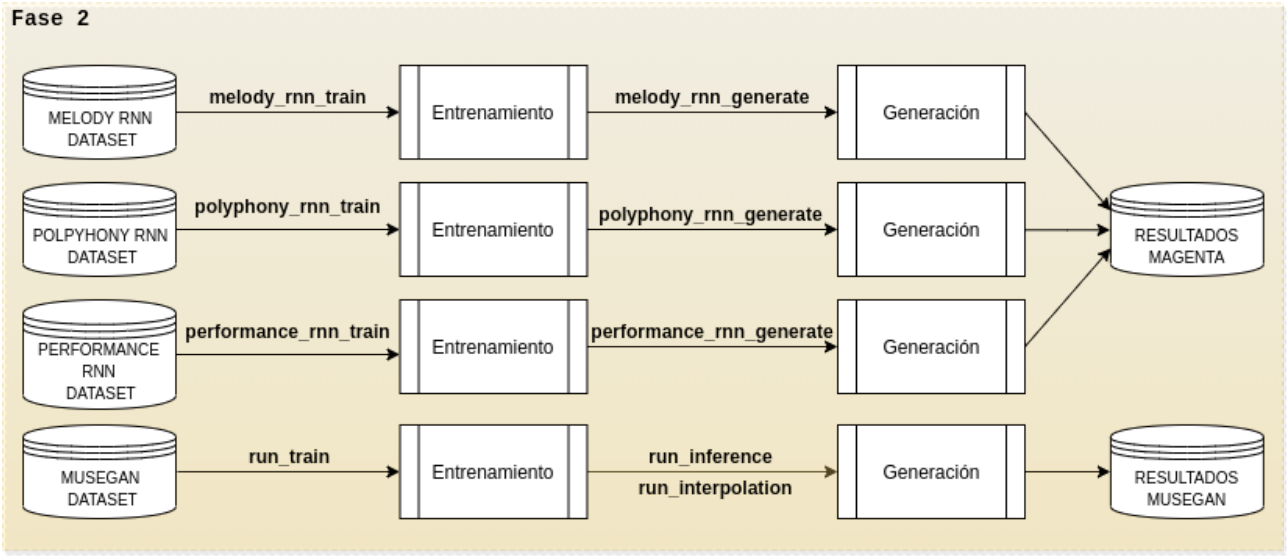


Figura 4.3: Fase 2 del experimento en detalle.

4.3. Fase 3: Extracción de características y obtención de resultados

Antes de comenzar el desarrollo de esta fase es necesario explicar cómo se ha determinado su estructura y por qué.

Las generaciones obtenidas en la **fase 2** parten de dos modelos distintos y por lo tanto su naturaleza no es la misma. Los modelos usados en Magenta tienen como objetivo generar melodías, mientras que MuseGAN por su parte genera composiciones en las que participan múltiples instrumentos. Si enfocamos esto desde un punto de vista musical, podemos ver el modelo MuseGAN como un grupo capaz de interpretar múltiples canciones desde cero, mientras que Magenta podemos verlo como un músico tocando un instrumento. Para mantener esta coherencia en los resultados finales debemos adaptar este enfoque, grupo frente a individuo, a la hora de aplicar las técnicas MIR mencionadas en el capítulo 3. Teniendo en cuenta este enfoque, proponemos la extracción de resultados de la siguiente forma:

Las generaciones obtenidas mediante **Magenta** se analizarán haciendo uso de la herramienta **Musicnn** así como de una encuesta pública que explicaremos más adelante en esta sección.

Las generaciones obtenidas mediante **MuseGAN** se analizarán a través de las herramientas **Musicnn** y **LSTM GClassifier**.

No tiene sentido aplicar la herramienta extractora de géneros a una melodía, ya que ésta no tiene por qué estar relacionada con un género en concreto, sin embargo, las melodías suelen representar una emoción, por lo que aplicar el etiquetado musical sobre éstas si nos puede proporcionar información útil. Las composiciones de MuseGAN no se incluyen en la encuesta a fin analizar éstas exclusivamente a partir de técnicas MIR.

A la hora de extraer características, lo primero que hemos de tener en cuenta es que las generaciones obtenidas están en formato MIDI y *pianoroll* por lo que no podemos, ni escucharlas, ni utilizarlas en las dos tecnologías MIR planteadas. Debemos por lo tanto convertir éstas a un formato que nos permita trabajar con ellas. Ambas tecnologías MIR nos permiten trabajar sobre los formatos de audio más comunes, **MP3** y **wav**. Entre estos dos elegimos *MP3* principalmente por limitaciones de espacio en el equipo, ya que debemos convertir tanto datos de entrenamiento como las generaciones obtenidas.

Conversión generaciones a formato MP3

Es importante mencionar que la transformación **manual** de un solo fichero MIDI a MP3 es una tarea sencilla que se puede llevar a cabo en cualquier herramienta online o DAW en cuestión de segundos, sin embargo, realizar esta tarea con más de 10000 ficheros supone un esfuerzo desmedido, por lo que debemos automatizar este proceso para realizarlo de forma local.

Para conseguir ésto necesitamos poder convertir un archivo MIDI a uno MP3 y, convertir un fichero *pianoroll* a MIDI.

Para convertir un fichero MIDI a un formato de audio, necesitamos utilizar una fuente de sonidos [68] que interprete la información existente en éste y la transforme a audio. Mediante la librería **fluidsynth** [69] somos capaces de aplicar una fuente de sonido a un MIDI y obtener audio en formato wav. Como ya se ha mencionada anteriormente, el formato wav tiene un tamaño superior al formato MP3. Como no podemos almacenar tanta información en nuestro equipo debemos encontrar una herramienta que nos permita transformar los ficheros wav a MP3 en tiempo de ejecución. Encontramos esta funcionalidad en la librería **pydub** [70], enfocada principalmente en el tratamiento de audio a través de los formatos más extendidos en la actualidad. Haciendo uso de estas dos librerías podemos automatizar la conversión de MIDI a MP3.

Podemos convertir los ficheros en formato *pianoroll* a MIDI gracias a una herramienta desarrollada por los autores de MuseGAN. Ésta herramienta es **pypianoroll** [71] y nos permite adaptar nuestros ficheros a distintos formatos de audio para trabajar fácilmente con ellos. Al ser una librería en python, también podemos automatizar el proceso de conversión de *pianoroll* a MIDI.

Una vez hemos comprobado que es posible automatizar las tareas de conversión planteadas, desarrollamos dos scripts python encargados de llevar a cabo los procesos descritos. Para el primer proceso nos basamos en la descripción que se hace en [72] sobre este mismo problema. El script desarrollado

en el artículo tiene ciertas limitaciones que debemos cubrir, por ejemplo, la incapacidad de transformar un fichero MIDI mediante *fluidsynth* a MP3, la incapacidad de trabajar con múltiples ficheros en una sola ejecución o la incapacidad de ejecutar esta tarea mediante múltiples hilos.

Para solucionar esta limitaciones hemos incluido los siguientes fragmentos de código dentro del script desarrollado en [72]

Código 4.2: Fragmento de código para recorrer todos los ficheros de un directorio en un formato concreto. En este caso, formato MIDI (.mid)

```
34 def findall_endswith(root):
35     """Traverse `root` recursively and yield all files ending with `postfix`"""
36     for dirpath, _, filenames in os.walk(root):
37         for filename in filenames:
38             if filename.endswith('.mid'):
39                 yield os.path.join(dirpath, filename)
```

En el código 4.2 recorremos recursivamente un directorio buscando todos los ficheros que existan con la extensión indicado en la línea 38. En nuestro caso, esta extensión corresponde a los ficheros MIDI. Este fragmento de código se ha extraído de uno de los scripts del la herramienta *lakh-pianoroll-dataset* [66].

Código 4.3: Configuración multihilo en python.

```
123         if CONFIG['multicore'] > 1:
124             joblib.Parallel(n_jobs=CONFIG['multicore'], verbose=5)(
125                 joblib.delayed(to_audio)(random.choice(sf2files), mid, out_dir, out_type)
126                 for mid in findall_endswith(midi_dir))
```

El código 4.3 nos permite ejecutar la tarea **to_audio** en paralelo dependiendo del número de núcleos de los que disponga nuestro procesador. Esto se determina a partir de la variable **CONFIG['multicore']**. En la línea 126 indicamos que esta tarea se debe realizar para todos los ficheros que se encuentren en el directorio indicado por la variable **midi_dir**. La función **to_audio** es la que encontramos en el código 4.4. Los parámetros que recibe se indican en la línea 125 del fragmento de código anterior. Éstos son:

- **sf2:** Ruta a un fichero del tipo fuente de sonidos.
- **midi_file:** Ruta al fichero MIDI a convertir.
- **out_dir:** Directorio dnde se almacenará el fichero convertido.
- **out_type:** Tipo de formato de audio al que convertir el fichero. Éste parámetro no acepta el formato MP3. Su valor por defecto es wav.

Es importante mencionar que, el primer parámetro, *sf2*, recibe una fuente de sonidos aleatoria desde un directorio en el que existen múltiples fuentes. Todas estas fuentes son distintas, respetan

los instrumentos existentes en los MIDIs utilizados y proporcionan un carácter distinto. Al hacer esta selección aleatoria le damos a cada composición un sonido y estilo diferente. Esto podría interpretarse como diferentes grupos interpretando piezas musicales cada uno en su estilo.

Código 4.4: Conversión MIDI a MP3.

```
55      fbase = os.path.splitext(os.path.basename(midi_file))[0]
56      out_file = out_dir + '/' + fbase + '.' + out_type
57      mp3_name = out_dir + '/' + fbase + ".mp3"
58      already_base = 'PopMidi_already/'
59      already_name = already_base + fbase + '.' + out_type
60
61      if os.path.isfile(out_file) or os.path.isfile(already_name):
62          randomNum = str(randint(0, 512))
63          out_file = out_dir + '/' + randomNum + "-" + fbase + '.' + out_type
64          mp3_name = out_dir + '/' + randomNum + "-" + fbase + ".mp3"
65
66          # Renombramos el midi para poder moverlo luego
67          midi_parts = midi_file.rsplit('.', 1)
68          midiAux1 = midi_parts[0] + "-" + randomNum + "."
69          midi_new_name = midiAux1 + midi_parts[1]
70          os.rename(midi_file, midi_new_name)
71          midi_file = midi_new_name
72
73      try:
74          subprocess.call(['fluidsynth', '-T', out_type, '-F', out_file, '-ni', sf2, midi_file])
75          shutil.move(midi_file, already_base)
76
77          song = AudioSegment.from_wav(out_file)
78          extract = song[15000:25000]
79
80          extract.normalize()
81          extract.export(mp3_name, format="mp3", bitrate="96k")
82          os.remove(out_file)
83      except:
84          print("No se ha podido transformar el fichero:", midi_file)
85          return
```

En la línea 74 se llama a la librería *fluidsynth* y se realiza la conversión MIDI a wav. Entre las líneas 77 y 81 se realiza la conversión de éste fichero wav a MP3 con diversas modificaciones. En la línea 77 establecemos el intervalo de tiempo que queremos guardar, en el código vemos el rango **15000:25000**, ésto nos indica que solamente guardaremos el audio existente entre los segundos 15 y 25. Posteriormente, en la línea 81 establecemos el *bitrate* de la pista a '96k', disminuyendo así su tamaño a cambio de perder un poco de calidad. Estas modificaciones se han realizado conforme a lo observado en distintos artículos y repositorios MIR.

La decisión de almacenar 10 segundos se tomó de acuerdo al estándar empleado en la web **musicinformationretrieval.com**. Ésta web se define así misma como *"una colección de materiales de instrucción para la recuperación de información musical (MIR)."*. Entre estos materiales encontramos

un ejemplo de reconocimiento de género en el que los fragmentos de audio a clasificar tienen una duración de 10 segundos. Por otra parte, la herramienta *Musicnn* que utilizaremos más adelante, utiliza por defecto, 3 segundos de audio.

La disminución del *bitrate* de '128k' a '96k' apenas altera la cantidad de información almacenada en nuestro fichero MP3, sin embargo, nos permite ahorrar mucho espacio en el equipo y aún así llevar a cabo la tarea de extracción de características satisfactoriamente.

Código 4.5: Conversión pianoroll (.npz) a MIDI.

```
28      # Type of experiment
29      if "inference" in src:
30          title = "inf_"
31      else:
32          title = "int_"
33
34      # Decision boundaries
35      if "bernoulli" in src:
36          title += "ber_"
37      elif "thresholding" in src:
38          title += "thr_"
39      else:
40          title += "bns_"
41
42      i = 1
43      for npz in findall_endswith(src):
44
45          multitrack = pypianoroll.Multitrack(npz)
46          title2 = 'gen_' + str(i) + '.mid'
47          multitrack.write(title+title2)
48
49          finalTitle = title + title2
50
51          result_path = os.path.join(dst, finalTitle)
52          shutil.move(finalTitle, result_path)
53          i += 1
```

El segundo proceso de conversión (*pianoroll* a MIDI) resulta muy sencillo gracias a la documentación existente en [71]. Empleamos el código 4.2 junto con el código 4.5 para automatizar este proceso.

Entre las líneas 29 y 40 asignamos el prefijo de los títulos que tendrán los ficheros que vamos a generar. Éste depende del tipo de de generación realizada y si el experimento es clásico o binario. A partir del bucle que comienza en la línea 43 realizamos la transformación y la asignación del nombre. En la línea 45 se lee el contenido del fichero *pianoroll*, en la 46 asigna el nombre y en la 47 se transforma. Con el formato que le hemos dado a los títulos evitamos escribir sobre el mismo fichero continuamente.

A partir de estos dos scripts podemos convertir nuestra colección de MIDIs a MP3 en una sola

ejecución. Una vez disponemos del conjunto de datos inicial y las generaciones obtenidas en MP3, pasamos a utilizar las técnicas MIR propuestas en el capítulo 3.

4.3.1. Etiquetado musical mediante Musicnn

Como ya mencionamos en el apartado 3.3, **Musicnn** nos permite aplicar un conjunto de técnicas MIR sobre un audio. Su funcionalidad principal reside en el **etiquetado musical**, técnica que aplicaremos sobre la colección de MP3s obtenidos a través de la conversión anteriormente descrita.

Al disponer de varios modelos pre-entrenados sobre datasets musicales ampliamente extendidos, la utilización de esta herramienta simple. Atendiendo a las instrucciones que encontramos en el repositorio mencionado en [53] necesitamos ejecutar un comando para extraer etiquetas de un fichero. A partir del código encargado de realizar ésto, creamos un nuevo script para automatizar el proceso de etiquetado musical.

Código 4.6: Etiquetado musical mediante Musicnn.

```
28  def extractTags(directory):
29
30      mtt_tags = {label: 0 for label in MTT_LABELS}
31      msd_tags = {label: 0 for label in MSD_LABELS}
32
33      for audio in findall_endswith(directory):
34          try:
35              audio_tags_mtt = top_tags(audio, model='MTT_musicnn', topN=10, input_length=6.0,
                    print_tags=print_tags)
36              audio_tags_msd = top_tags(audio, model='MSD_musicnn_big', topN=10,
                    input_length=6.0, print_tags=False)
37
38              for tag_mtt in audio_tags_mtt:
39                  mtt_tags[tag_mtt] += 1
40              for tag_msd in audio_tags_msd:
41                  msd_tags[tag_msd] += 1
42          except:
43              print ("No_se_ha_podido_transformar_el_fichero", audio)
44
45      return mtt_tags, msd_tags
```

En el código 4.6 detallamos la función **extractTags** que recibe como parámetro el directorio en el que está almacenada nuestra colección de MP3s. Entre la línea 30 y 36 configuramos el proceso que vamos a llevar a cabo. Exceptuando las líneas 32 y 33, encargadas de inicializar unos diccionarios dónde almacenaremos las ocurrencias de cada etiqueta, el resto de líneas hacen referencia a los parámetros de la la función **top_tags**. Ésta función es la encargada de devolver las etiquetas más representativas del audio que recibe el proceso. Los parámetros de esta función son:

- **audio:** Ruta al fichero MP3 al que aplicaremos el proceso de etiquetado musical.

- **model:** Representa el modelo seleccionado en el etiquetado musical. Existen múltiples opciones para este parámetro.
  - **MTT_musicnn:** Modelo pre-entrenado en el dataset *MagnaTagATune* enfocado al etiquetado musical.
  - **MTT_vgg:** Modelo pre-entrenado en el dataset *MagnaTagATune* enfocado a la extracción de características y el transpaso de aprendizaje.
  - **MSD_musicnn:** Modelo pre-entrenado a partir del *Million Song Dataset* enfocado al etiquetado musical.
  - **MSD_musicnn_big:** Modelo pre-entrenado a partir del *Million Song Dataset* enfocado al etiquetado musical. Se diferencia de *MSD_musicnn:* en la estructura de la CNN aplicada, contando el modelo **big** con un mayor número de filtros y de unidades de *pooling*.
  - **MSD_vgg:** Modelo pre-entrenado a partir del *Million Song Dataset* enfocado a la extracción de características y el transpaso de aprendizaje.
- **topN:** Número de etiquetas a extraer. Se extraerán aquellas que son más representativas en el audio.
- **input_length:** Longitud en segundos del audio a analizar. Por defecto su valor es '3.0'.
- **print_tags:** Si está activado (**True**), muestra trazas sobre las etiquetas extraídas para cada audio.

Nuestro objetivo es obtener la información más representativa y de alto nivel posible de cara al análisis de resultados, por lo que, extraemos etiquetas mediante los modelos **MTT_musicnn** y **MSD_musicnn_big**. Cambiamos el valor del parámetro *input_length* a '6.0' para que los audio analizados contengan más información, modificamos *topN* para extraer solamente las 10 etiquetas más representativas del audio y desactivamos las trazas para agilizar la ejecución del proceso.

Una vez hemos extraído todos estos datos de nuestra colección de MP3s completa (Conjunto de datos original y generaciones), continuamos con la etapa de reconocimiento de género.

4.3.2. Reconocimiento de género mediante LSTM GClassifier

Esta tarea se realiza conforme a la documentación existente en [56]. Del mismo modo que en las secciones anteriores, necesitamos automatizar este proceso.

Para obtener el género de un fichero MP3 utilizamos un script (**predict_example.py**) encargado de extraer las características del audio que posteriormente recibirá el clasificador. Podemos incluir el código 4.2 para trabajar directamente sobre un directorio, obteniendo así la automatización. Los géneros predichos durante la ejecución se almacenan en un fichero de texto con el siguiente formato:

$AUDIO_SOURCE;$AUDIO_NAME;$GENRE

Dónde **$AUDIO_SOURCE** indica el origen del audio, es decir, si forma parte del conjunto de datos inicial o si es una generación. Este valor se añade como un nuevo parámetro para el script. **$AUDIO_NAME** indica el nombre del fichero y **$GENRE** representa el género obtenido.

Esta herramienta no dispone de configuraciones que cambien su funcionalidad o estructura.

Una vez hemos extraído los géneros de nuestra colección de MP3, damos por finalizada la fase de extracción de resultados a partir de técnicas MIR y continuamos con el desarrollo de la encuesta.

4.3.3. Encuesta musical

Esta encuesta se ha realizado a través de la herramienta **Google Forms** por su fácil integración con **Google Drive**, herramienta dónde almacenaremos las melodías que formarán parte de la encuesta.

Como ya introdujimos al comienzo de esta sección, se ha desarrollado una encuesta pública que pretende analizar desde un punto de vista humano las diferencias que somos capaces de encontrar entre una composición real y una composición artificial. La música, al ser un campo tan amplio y subjetivo puede ser interpretada de forma distinta por cada persona, por lo que es interesante reflejar esto a través de la encuesta. Explicaremos a continuación la estructura y decisiones tomadas en el diseño de la misma.

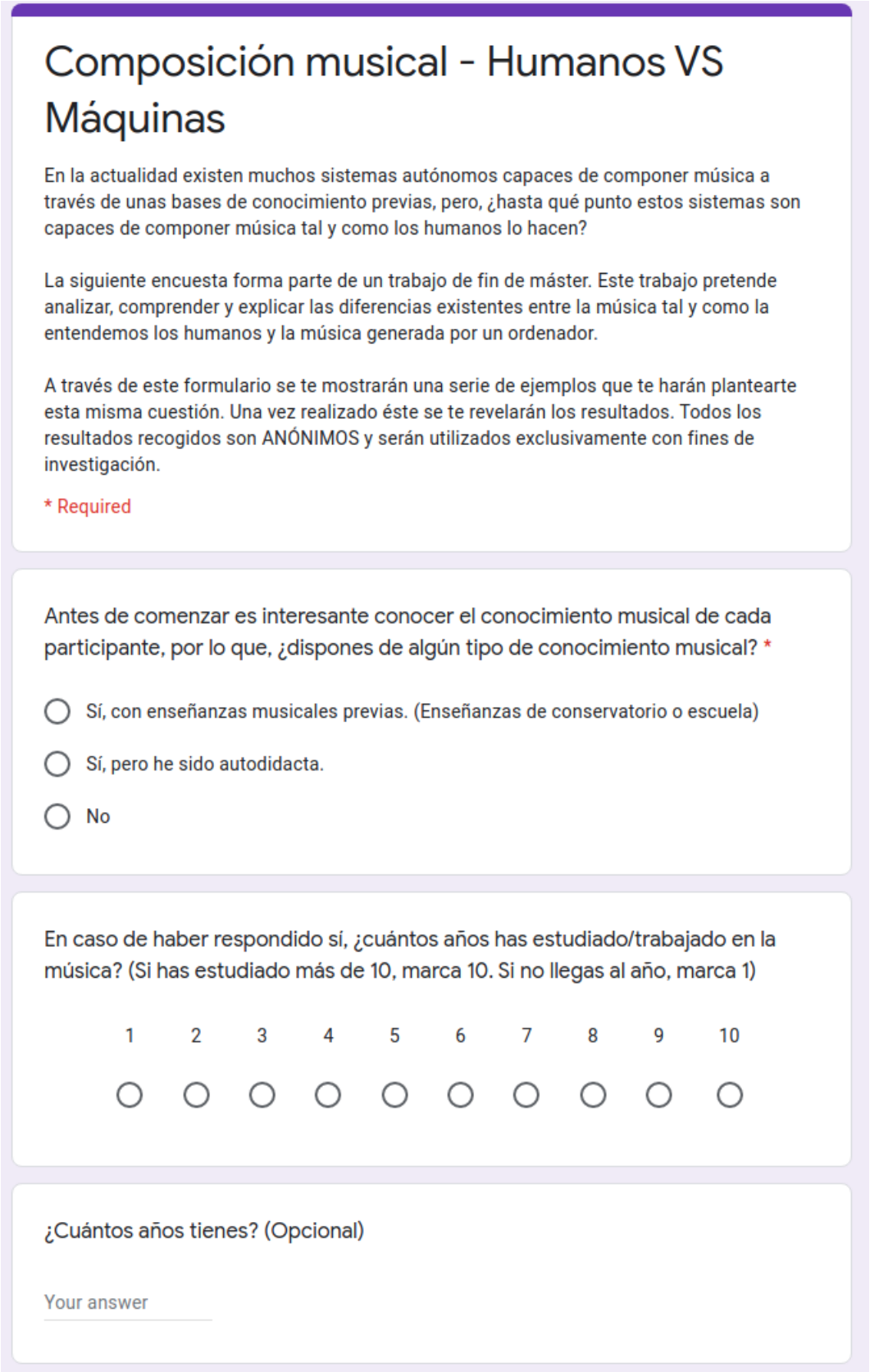
La encuesta está compuesta por tres secciones distintas. La primera sirve de introducción y pretende recoger datos que nos puedan proporcionar información sobre los conocimientos musicales de cada usuario. Al ser una encuesta anónima, los únicos datos obligatorios son los **conocimientos musicales** del usuario, permitiéndonos de este modo clasificarlos en distintos grupos. Además de estos, se recoje la edad de forma opcional con el objetivo de observar si existe correlación entre esta y las puntuaciones. Se presenta la primera sección en la figura 4.4

A partir de los datos que recogemos, podemos crear distintos grupos de usuarios a observar. Estableciendo como criterio único el conocimiento musical de cada usuario, divimos a éstos en los siguientes grupos:

- **Grupo 1:** Usuarios que han indicado que no disponen de conocimientos musicales.
- **Grupo 2:** Usuarios que han indicado que sí disponen de conocimientos musicales y tienen entre 0 y 3 años de experiencia.
- **Grupo 3:** Usuarios que han indicado que sí disponen de conocimientos musicales y tienen entre 4 y 7 años de experiencia.
- **Grupo 4:** Usuarios que han indicado que sí disponen de conocimientos musicales y tienen más de 7 años de experiencia.

A pesar de que se realizará un análisis global de los resultados, estos grupos nos permitirán obtener conclusiones adiciones sobre el experimento. Se plantearán observaciones que muestren el comportamiento del factor edad sobre las puntuaciones dadas, sin embargo no consideramos que este dato pueda proporcionar resultados igual de fiables que aquellos obtenidos mediante conocimientos musicales.

La segunda sección está compuesta por un conjunto de diez melodías generadas mediante los



Composición musical - Humanos VS Máquinas

En la actualidad existen muchos sistemas autónomos capaces de componer música a través de unas bases de conocimiento previas, pero, ¿hasta qué punto estos sistemas son capaces de componer música tal y como los humanos lo hacen?

La siguiente encuesta forma parte de un trabajo de fin de máster. Este trabajo pretende analizar, comprender y explicar las diferencias existentes entre la música tal y como la entendemos los humanos y la música generada por un ordenador.

A través de este formulario se te mostrarán una serie de ejemplos que te harán plantearte esta misma cuestión. Una vez realizado éste se te revelarán los resultados. Todos los resultados recogidos son ANÓNIMOS y serán utilizados exclusivamente con fines de investigación.

* Required

Antes de comenzar es interesante conocer el conocimiento musical de cada participante, por lo que, ¿dispones de algún tipo de conocimiento musical? *

- ◯ Sí, con enseñanzas musicales previas. (Enseñanzas de conservatorio o escuela)
- ◯ Sí, pero he sido autodidacta.
- ◯ No

En caso de haber respondido sí, ¿cuántos años has estudiado/trabajado en la música? (Si has estudiado más de 10, marca 10. Si no llegas al año, marca 1)

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

¿Cuántos años tienes? (Opcional)

Your answer

Figura 4.4: Primera sección de la encuesta.

modelos de Magenta. De estas diez, **seis** pertenecen al modelo Melody RNN, **dos** al modelo Performance RNN y las **dos** restantes al modelo Polyphony RNN. De las seis generaciones obtenidas a partr del modelo Melody RNN, **tres** de ellas pertenecen a la configuración **Lookback** y las restantes a la configuración **Attention**. Los MIDIs pertenecientes a estas generaciones se han procesado en la plataforma de audio digital FL Studio [1]. A fin de no modificar las generaciones, los únicos parámetros modificados han sido el tempo, el instrumento que interpreta las melodías y la longitud de éstas, que se ha reducido a ocho compases de los dieciséis originales. El objetivo de este procesamiento es darle a cada melodía un carácter distinto que plantee distintas perspectivas a los usuarios de cara a evaluar éstas.

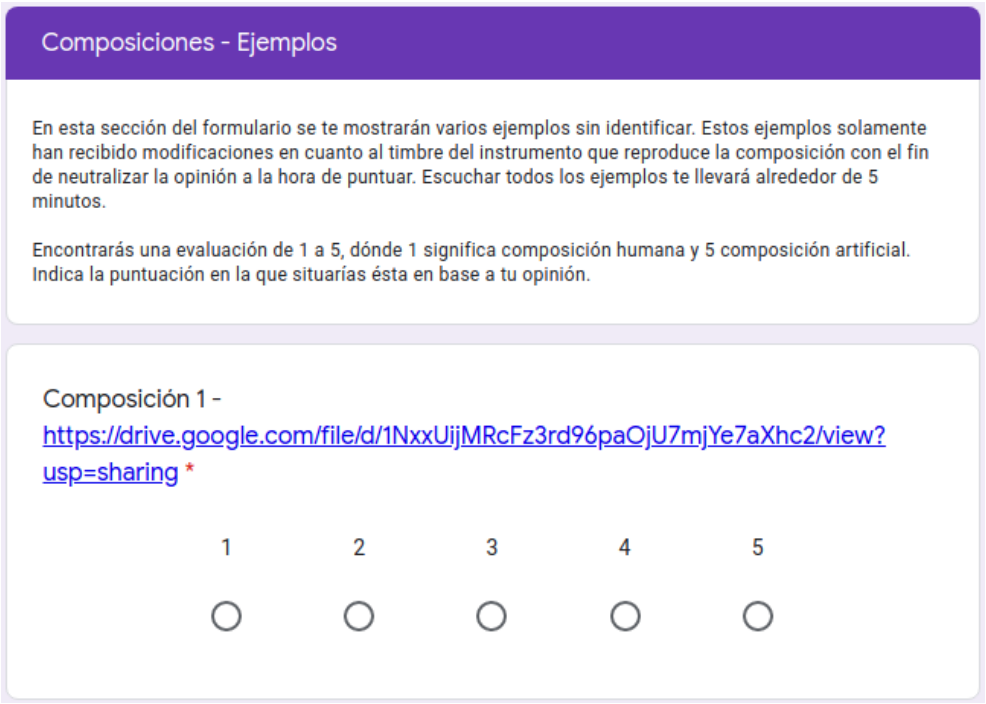


Figura 4.5: Segunda sección de la encuesta. Ejemplo de una melodía.

La figura 4.5 muestra un ejemplo de esta sección con una composición de las empleadas para consultar a los usuarios.

La evaluación de las melodías se realiza mediante la escala de Likert [73]. Ésta nos permite plantear las posibles respuestas de cada usuario a través de un intervalo en el que los extremos representan conceptos opuestos. En nuestro caso, uno de los extremos es el valor '1' y representa que la melodía en cuestión ha sido compuesta por un humano, mientras que el otro extremo es el valor '5' e indica que la melodía ha sido generada a través de un computador. Entre estos extremos encontramos los valores '2', '3' y '4', que permiten a los usuarios ajustar sus respuestas de una forma más precisa que con una respuesta del tipo sí/no. Debido a las limitaciones de la herramienta *Google Forms*, las melodías se incluyen a partir de enlaces a *Google Drive*. Una vez se han evaluado las diez melodías propuestas se

[1] Página web FL Studio: https://www.image-line.com/fl-studio/

permite avanzar a la siguiente sección.

La última sección informa a los usuarios de que ninguna de las composiciones es humana, agradeciendo su participación. Además plantea, de forma opcional, dos cuestiones más. La primera, cuestiona al usuario sobre los géneros musicales que ha observado en las melodías propuestas. La segunda, permite rellenar un cuadro de texto libre donde opinar sobre el experimento. Gracias a estas cuestiones opcionales, podemos obtener reflexiones de los usuarios sobre el concepto por el cual se plantea la encuesta. Se muestra esta sección de la encuesta en la figura 4.6

Con el objetivo de obtener más resultados, la encuesta se tradujo al inglés y se publicó en múltiples foros de la red social [2]. Todos los foros en los que se publicó ésta están relacionados con la inteligencia artificial y sus aplicaciones. Éstas publicaciones y los cuestionarios los encontramos en los siguientes enlaces:

- https://www.reddit.com/r/artificial/comments/i2uzmk/how_we_humans_differentiate_human_music_to/
- https://www.reddit.com/r/MediaSynthesis/comments/i2uynm/how_we_humans_differentiate_human_music_to/
- https://www.reddit.com/r/MachineLearning/comments/i30oux/r_how_we_humans_differentiate_human_music_to/
- **Español:** https://forms.gle/pfM4DSWCAdDLjZky8
- **Inglés:** https://forms.gle/dxLLGPwQrX6uaMKY9

Los resultados obtenidos se analizarán en el siguiente capítulo junto a los datos extraídos en las anteriores secciones.

[2] Página web reddit: https://www.reddit.com/

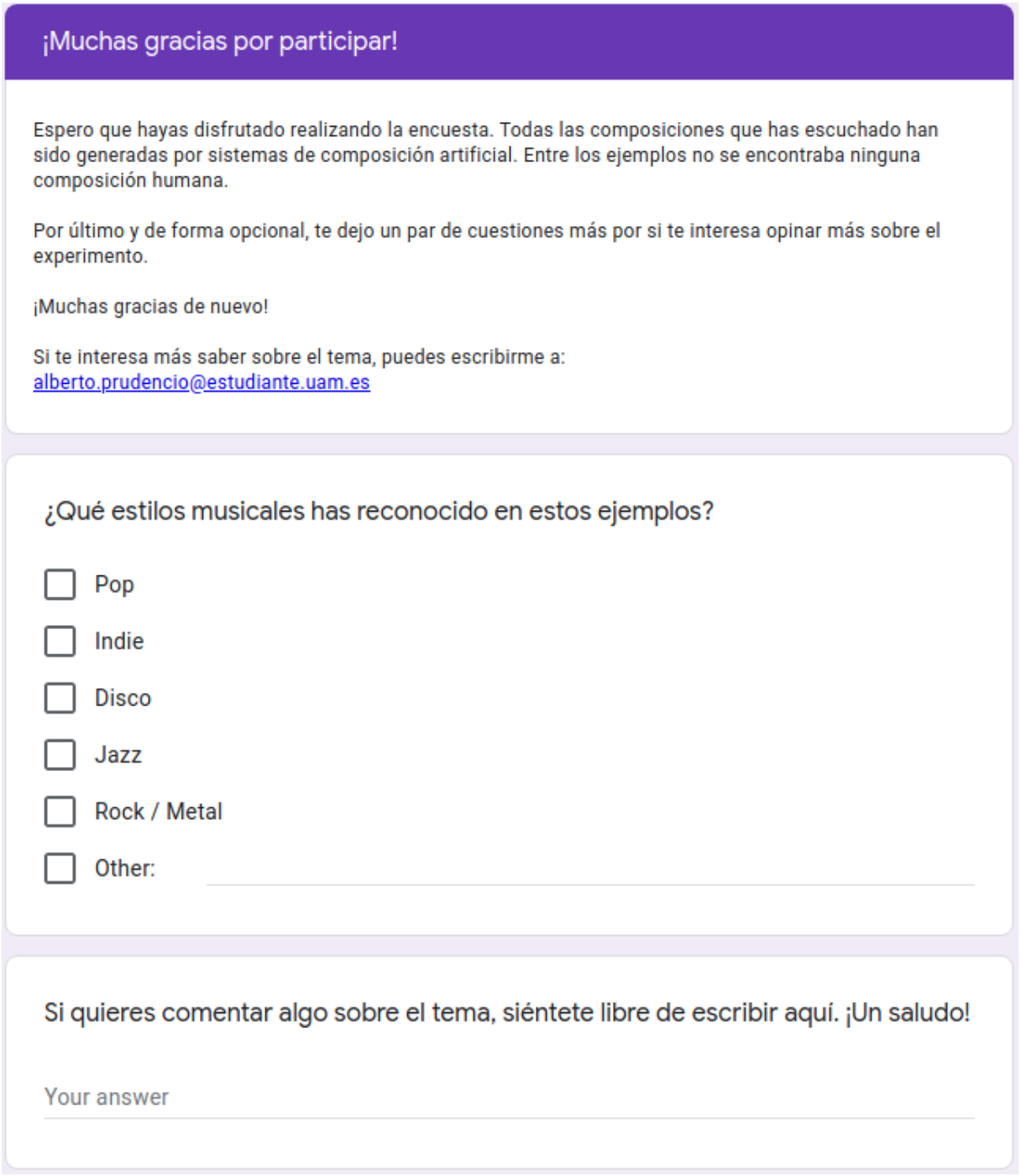


Figura 4.6: Tercera sección de la encuesta.

Ampliando la fase 3 de la figura 4.1 podemos detallar la arquitectura completa como se muestra en la figura siguiente:

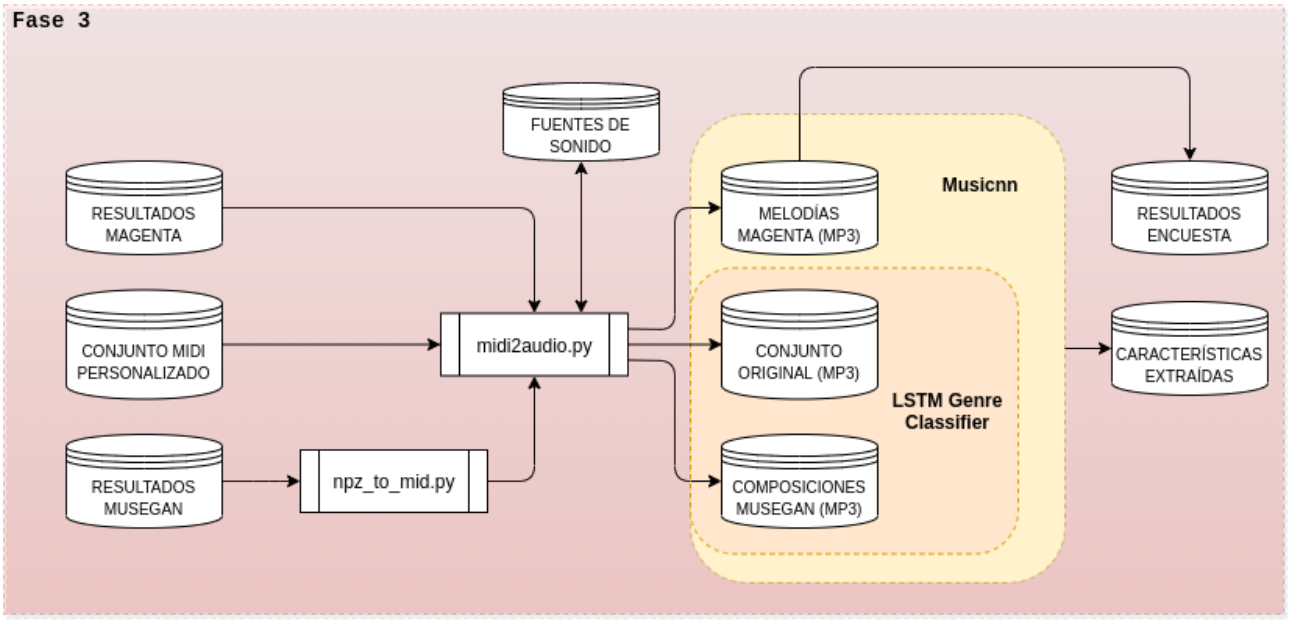


Figura 4.7: Fase 3 del experimento en detalle.

5

Análisis de los resultados

A lo largo de este capítulo se realizará un análisis de los resultados obtenidos. El objetivo **O-5**, tal y cómo se mencionó en el apartado 1.2, es comprender hasta qué punto las generaciones obtenidas a lo largo del experimento se asemejan al conjunto de datos original. Se observarán las diferencias y similitudes existentes entre ambos, así como las diferencias entre los distintos modelos usados.

Del mismo modo que en la fase 3 del capítulo anterior, dividiremos el análisis en tres secciones. La primera, enfocada al **etiquetado musical**, la segunda, al **reconocimiento de géneros musicales** y la tercera a la **encuesta musical**. Para ello, se utilizarán histogramas y el error cuadrático medio. De esta forma, podremos concluir tanto visualmente como teóricamente qué resultados se asemejan más a los devueltos por el conjunto original.

El **error cuadrático medio** o MSE (Mean Squared Error) por sus siglás en inglés, es un estimador que nos permite obtener el promedio de las diferencias cuadráticas existente entre el valor original y el estimado. La fórmula para realizar esto es la siguiente:

$$ECM(MSE) = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{5.1}$$

Dónde n es el número de observaciones, $\hat{Y}_i$ es la estimación y Y_i el valor real observado.

Para ajustar esto al experimento realizado, consideramos el estimador, los datos devueltos por las técnicas MIR aplicadas al **conjunto original de datos**, y las estimaciones de los datos obtenidos a partir de las técnicas MIR aplicadas a las **generaciones obtenidas** en la fase 2 del capítulo anterior.

Usamos histogramas para representar visualmente nuestros resultados debido al formato en el que hemos almacenado éstos. En el caso del etiquetado musical, cada fichero analizado devuelve un conjunto de etiquetas, por lo que, para cada modelo utilizado almacenamos las frecuencias de las etiquetas obtenidas en formato **CSV**. Por ejemplo, las etiquetas MTT obtenidas tienen el siguiente formato:

rock;pop;alternative;...;sad;House;happy
38;36;2;...;0;0;0

Cuadro 5.1: Formato de almacenado de etiquetas musicales.

La primera fila representa las etiquetas y la segunda la frecuencia de éstas en los datos propocionados. Respecto a los géneros obtenidos, el formato es el mismo, cambiando las etiquetas por géneros.

Por último, con el objetivo de poder hacer referencia a los conjuntos de datos analizados en las siguientes secciones, indicaremos las dimensiones de cada uno y sus denominaciones:

- **ORIGINAL:** Colección de MP3s obtenidos a partir del conjunto de datos original. Está formado por 13977 ficheros. El tamaño del conjunto original era de 14109 ficheros, sin embargo, debido a problemas en la conversión de MIDI a MP3 algunos ficheros se han descartado.
- **CLÁSICO:** Colección de MP3s obtenidos a través del experimento clásico en MuseGAN. Está formada por 160 composiciones, 80 de ellas obtenidos a través de **muestreo de Bernouilli** y 80 de ellas obtenidos a través de **umbralización dura**.
- **BINARIO:** Colección de MP3s obtenidos a través del experimento binario en MuseGAN. Está formada por 80 composiciones.
- **MAGENTA:** Colección de MP3s obtenidos a través de los distintos modelos de Magenta. Está formada por 170 melodías.

Debido a las diferencias en las dimensiones de éstos, debemos transformarlos a una misma escala. Para ello, dividimos la frecuencia de cada género/etiqueta por el número total de ejemplos proporcionado y obtenemos una escala entre los valores ’0’ y ’1’ que representa el porcentaje de aparición de cada una de estas características en un conjunto de datos. Gracias a esta conversión los resultados serán más fáciles de obtener y sencillos a la hora de analizar.

5.1. Análisis etiquetado musical

El etiquetado musical, en sus dos modalidades, MTT y MSD, se ha aplicado sobre todos los conjuntos de datos. Primero, analizaremos la modalidad MTT, centrada en el etiquetado a partir del dataset *MagnaTagATune* y después la modalidad MSD, centrada en el etiquetado a partir del dataset *Million Song Dataset*.

Todas las etiquetas con un porcentaje de aparición inferior al 10 % serán eliminadas de las gráficas a fin de facilitar la visualización de éstas. Los gráficos completos se encuentran en el apéndice B. De cara a la obtención del MSE, estos datos sí se tendrán en cuenta.

Análisis etiquetado musical con dataset MagnaTagATune (MTT)

La siguiente gráfica nos muestra una comparación de las etiquetas obtenidas por los distintos conjuntos de datos para el modelo MTT.

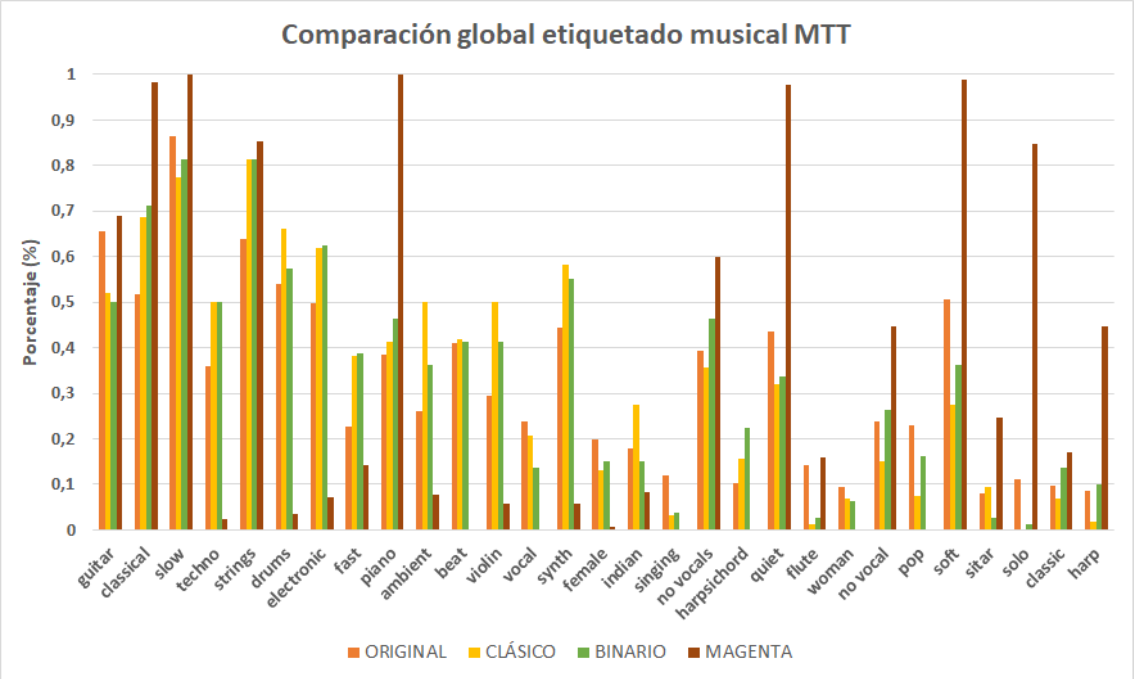


Figura 5.1: Comparación global en etiquetado musical MTT.

El MSE de cada modelo respecto al conjunto original es el siguiente:

ERROR	CLÁSICO	BINARIO	MAGENTA
MSE	0,00936	**0,00657**	0,0633

Tabla 5.1: Error cuadrático medio en etiquetado musical MTT.

Atendiendo al MSE obtenido, el modelo que más se asemeja al conjunto de datos original en este caso es el experimento binario. El experimento clásico tiene un MSE ligeramente superior, mientras que Magenta obtiene un MSE mucho mayor, considerándose por lo tanto, el modelo que menos se asemeja al conjunto original. Teniendo en cuenta que las composiciones del conjunto de datos original contienen múltiples instrumentos, es lógico pensar que, debido a la naturaleza de las generaciones de cada modelo, las melodías de Magenta siempre se asemejarán menos que las composiciones de MuseGAN.

Atendiendo al histograma, lo primero que nos llama la atención son las etiquetas en las que Magenta obtiene más de un 90 % de ocurrencia. Éstas son: *classical, slow, piano, quiet y soft*. Teniendo en cuenta que las melodías de Magenta solo contienen un piano (monofónico o polifónico) y tienen un tempo fijo de 110 BPM, es lógico haber obtenido las etiquetas *piano* y *slow*. Ya que no se han configurado más parámetros que los mencionados, consideramos el resto de etiquetas **descriptivas**.

A fin de obtener más información sobre las diferencias entre los modelos y el conjunto original, mostramos las diferencias cuadráticas para cada etiqueta. No mostramos las etiquetas de Magenta ya que son aquellas que más diferencia muestran con los datos originales.

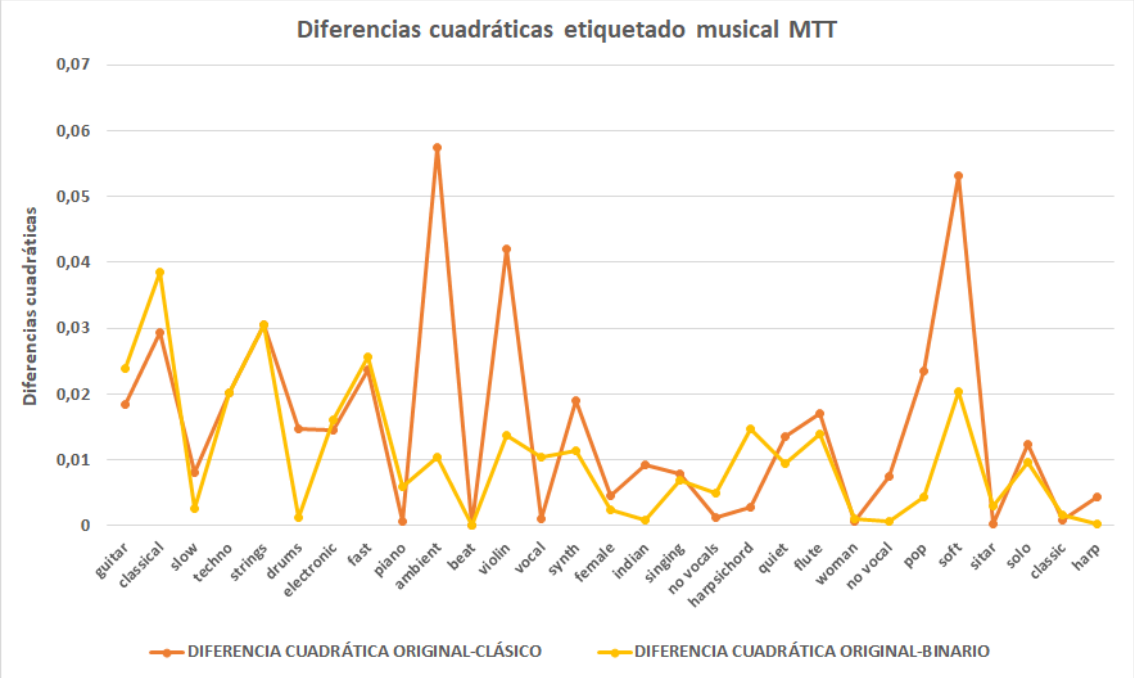


Figura 5.2: Diferencias cuadráticas en etiquetado musical MTT.

A pesar de que los MSE obtenidos para los experimentos clásico y binario son muy parecidos, gracias a este gráfico podemos observar las diferencias exactas de cada experimento con el original. Las únicas diferencias notables las podemos observar en las etiquetas *ambient, violin y soft*, dónde el experimento clásico se aleja más de los datos originales. A pesar de ésto, ambos modelos proporcionan un resultado muy similar, manteniendo prácticamente las mismas diferencias cuadráticas con el conjunto original en cada etiqueta.

En cuanto a las similitudes, vemos que las etiquetas con menos diferencia cuadrática son *slow, beat, woman, sitar y classic*. Dos de éstas, coinciden con las etiquetas de Magenta con ocurrencia superior al 90 %, por lo que podemos afirmar que estas características del conjunto original se han mantenido en las generaciones por encima del resto.

Aquellas etiquetas con un porcentaje de ocurrencia menor al 10 % son las siguientes: *rock, opera, male, vocals, loud, male vocal, man, choir, voice, new age, dance, female vocal, beats, cello, no voice, weird, country, metal, female voice, choral*. Algunas de éstas solo se encuentran en el conjunto de datos original, mientras que otras solo existen en las generaciones. A pesar de ésto, debido a su baja frecuencia no aportan más información que la ausencia de estas características en todos los conjuntos.

Análisis etiquetado musical con dataset Million Song Dataset (MSD)

Para analizar el conjunto de etiquetas perteneciente al *Million Song Dataset* seguiremos el mismo procedimiento que en el análisis previo. Mostramos por lo tanto, una gráfica comparativa de las etiquetas obtenidas en los distintos conjuntos de datos para el modelo MSD.

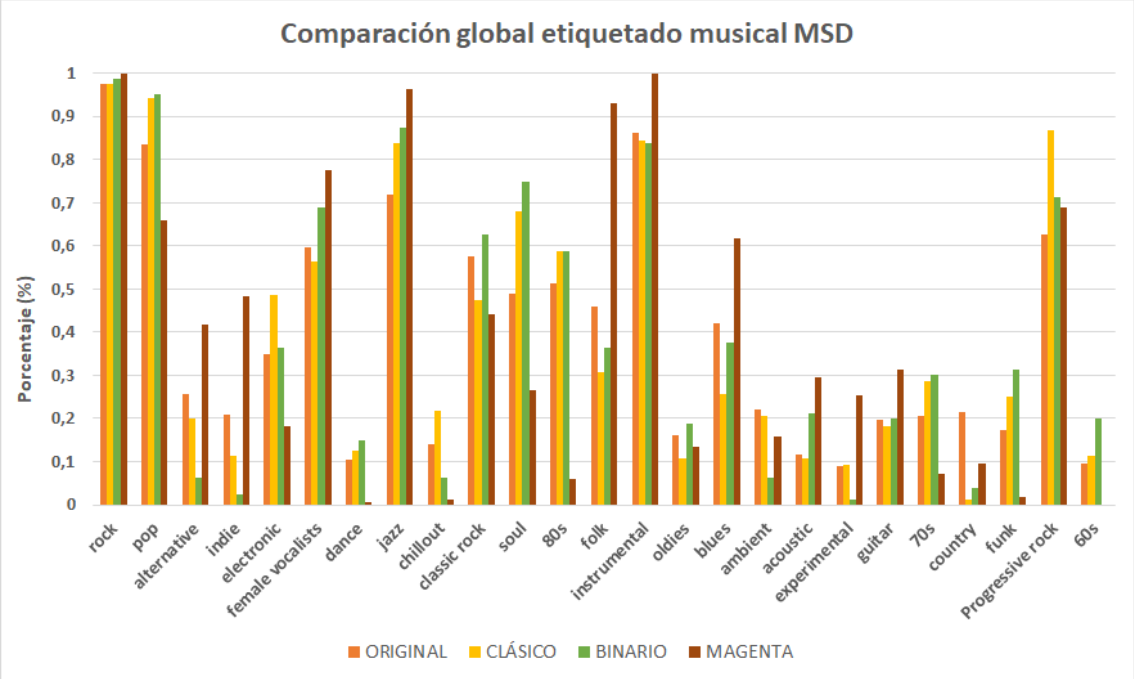


Figura 5.3: Comparación global en etiquetado musical MSD.

El MSE de cada modelo respecto al conjunto original es el siguiente:

ERROR	CLÁSICO	BINARIO	MAGENTA
MSE	**0,00583**	0,00697	0,0197

Tabla 5.2: Error cuadrático medio en etiquetado musical MSD.

Los resultados obtenidos en el MSE son similares a los del modelo MTT, sin embargo, esta vez el que más se asemeja al conjunto de datos original es el clásico. La diferencia entre los dos modelos de MuseGAN en esta ocasión es muy pequeña, por lo que han obtenido resultados muy similares. Magenta disminuye su MSE pero sigue siendo el modelo que más diferencias presenta.

Respecto al histograma, podemos ver como Magenta esta vez se distribuye más a lo largo del conjunto de etiquetas en vez de centrarse solamente en un subconjunto de éstas. Ésto, junto a su menor MSE nos indica un mejor ajuste respecto a los datos originales. Los experimentos de MuseGAN en esta ocasión son muy próximos, por lo que, a fin de obtener más información sobre éstos, mostramos las diferencias cuadráticas para cada etiqueta.

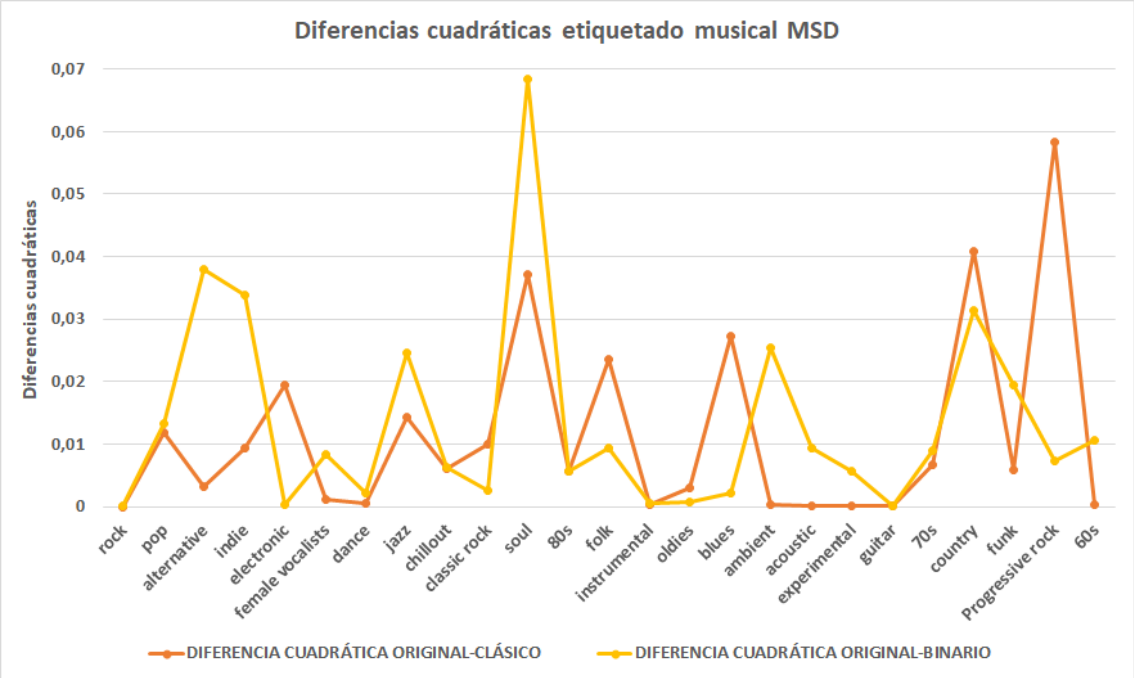


Figura 5.4: Diferencias cuadráticas en etiquetado musical MSD.

Mediante la figura 5.4 podemos observar fácilmente qué información se ha perdido en las generaciones y cual se ha mantenido. En este caso, las diferencias más notables las podemos observar en las etiquetas *soul y country*, mientras que las mayores similitudes se encuentran en las etiquetas *rock, dance, instrumental, oldies y guitar*. Exceptuando la etiqueta *dance*, Magenta comparte también estas diferencias y similitudes.

Las etiquetas descartadas son las siguientes: *00s, alternative rock, beautiful, metal, male vocalists, indie rock, Mellow, electronica, 90s, chill, punk, hard rock, female vocalist, Hip-Hop, party, easy-listening, sexy, catchy, electro, heavy metal, rnb, indie pop, sad, House y happy*. Ninguna de éstas tiene relevancia, ni en el conjunto original, ni en los distintos conjuntos de generaciones por lo que, del mismo modo que en el modelo MTT, estas etiquetas nos indican la ausencia de estas características en nuestros datos.

Observaciones

Gracias a su bajo MSE, los conjuntos procedentes de MuseGAN han demostrado ser aquellos que más información retienen del conjunto de datos original. Para el modelo MTT, el experimento binario obtiene mejores resultados, mientras que, para el modelo MSD el experimento clásico es el que mejores resultados ofrece. Aunque en ambos casos haya un caso favorable, ambos experimentos obtienen resultados muy similares por lo que parecen ofrecer prácticamente el mismo comportamiento ante estas técnicas MIR.

El conjunto de Magenta es aquel que menos se asemeja al original, sin embargo, como ya hemos

explicado antes, ésto es debido a las múltiples diferencias entre la naturaleza de éste y la del conjunto original. A pesar de ésto, el MSE obtenido y los histogramas nos demuestran que muchas de las características presentes en el conjunto de datos original existen también en éste.

Mediante las diferencias cuadráticas hemos podido ver aquellas etiquetas que más se han preservado en las generaciones respecto al conjunto original, siendo éstas: *slow, beat, woman, sitar y classic* para el modelo MTT y *rock, dance, instrumental, oldies y guitar* para el modelo MSD. No existe ninguna etiqueta que cuente con una diferencia cuadrática muy alta, por lo que, generalmente la información del conjunto de datos original se ha preservado.

En cuanto a los modelos de etiquetado musical, es curioso cómo el modelo MTT tiene entre sus etiquetas descartadas la etiqueta *rock*, mientras que en el modelo MSD todos los conjuntos superan el 95 % de frecuencia. Diferencias similares podemos verlas también en etiquetas como *pop, female vocalists o electronica*. Estas diferencias posiblemente se deban a la mayor capacidad del modelo MSD, tal y como explicamos en el apartado 4.3.1.

Terminado el análisis sobre el etiquetado musical, continuamos con el análisis de los resultados obtenidos en la técnica MIR de reconocimiento de género.

5.2. Análisis reconocimiento de géneros

Para realizar este análisis, se ha prescindido del conjunto de datos de Magenta, tal y como se indicó al comienzo del apartado 4.3.

Siguiendo el mismo procedimiento que en la sección anterior, obtendremos el histograma correspondiente, el MSE de los modelos utilizados y compararemos las diferencias cuadráticas. Los histogramas en esta ocasión no prescindirán de ningún valor por su frecuencia de aparición ya que solo existen ocho géneros y no dificultan la visualización de éstos. Podemos observar el histograma de géneros musicales en la figura 5.5.

El MSE de cada modelo respecto al conjunto original es el siguiente:

ERROR	CLÁSICO	BINARIO
MSE	0,0205	**0,0155**

Tabla 5.3: Error cuadrático medio en en reconocimiento de género.

El conjunto de datos con menor MSE es el binario, por lo que ésto nos indica que es el que menos diferencias guarda respecto al conjunto original. El conjunto clásico ofrece un resultado ligeramente peor, sin embargo, debido a la proximidad de los valores obtenidos podríamos decir que ambos conjuntos presentan prácticamente el mismo comportamiento ante esta técnica MIR.

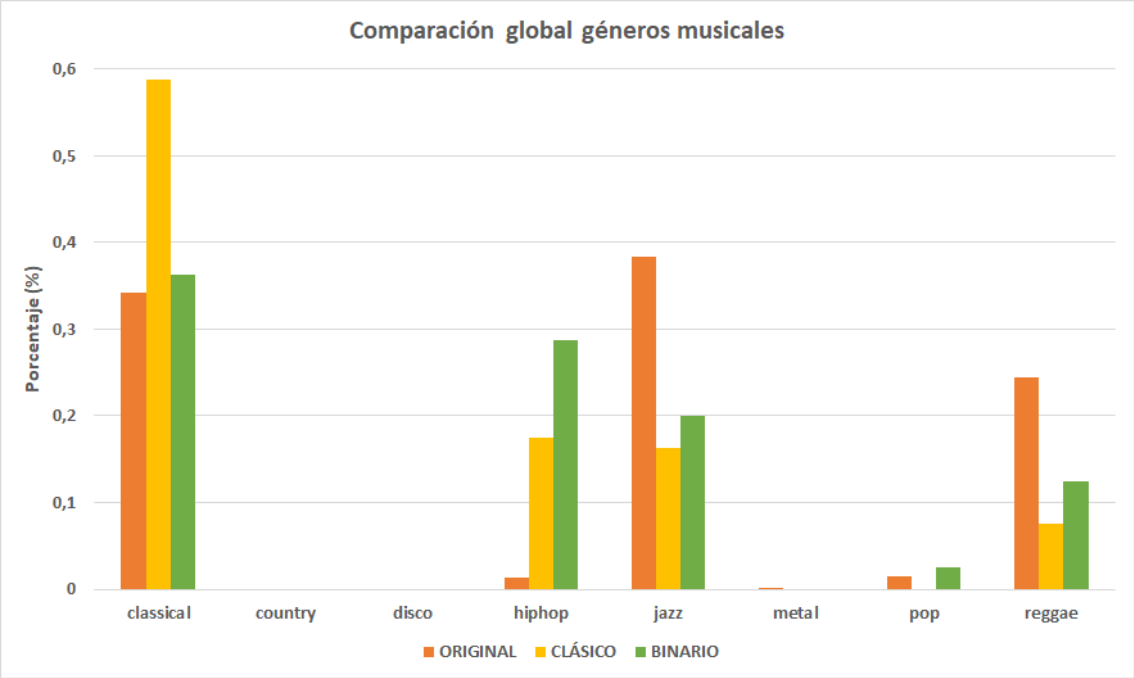


Figura 5.5: Comparación global en reconocimiento de género.

Observando el histograma, vemos que los géneros *country y disco* no están presentes en los resultados, y que los géneros *metal y pop* prácticamente son inexistentes. Otro detalle que nos sorprende es la desaparición de los géneros *rock y blues*. Ambos géneros forman parte del **GTZAN dataset** pero parecen no estar contemplados en la tarea de reconocimiento. Respecto al resto de géneros podemos observar mejor las diferencias entre los modelos y el conjunto de datos original mediante las diferencias cuadráticas en la figura 5.6.

Ambos conjuntos parecen preservar la información suficiente para obtener los mismos géneros que el conjunto original. Observamos que, a excepción del conjunto binario en el género *classical*, para el resto de géneros se obtienen diferencias mucho más altas. Las mayores diferencias se encuentran en los géneros *classical y jazz*, para el conjunto clásico y *hip-hop y jazz* para el conjunto binario.

Observaciones

La exclusión de los géneros *rock y blues* resulta en un aspecto muy negativo que, hasta este punto no se había descubierto.

Tanto el histograma como el MSE obtenido demuestran que existen múltiples diferencias entre los conjuntos de generaciones respecto al conjunto original. A pesar de que todos los conjuntos comparten los mismos géneros, las diferencias entre las frecuencias de éstos nos indica que no se ha preservado mucha información. El ejemplo más claro de ésto lo vemos en el género *hiphop*, sin embargo, también ocurre en menor medida para los géneros *jazz y reggae*.

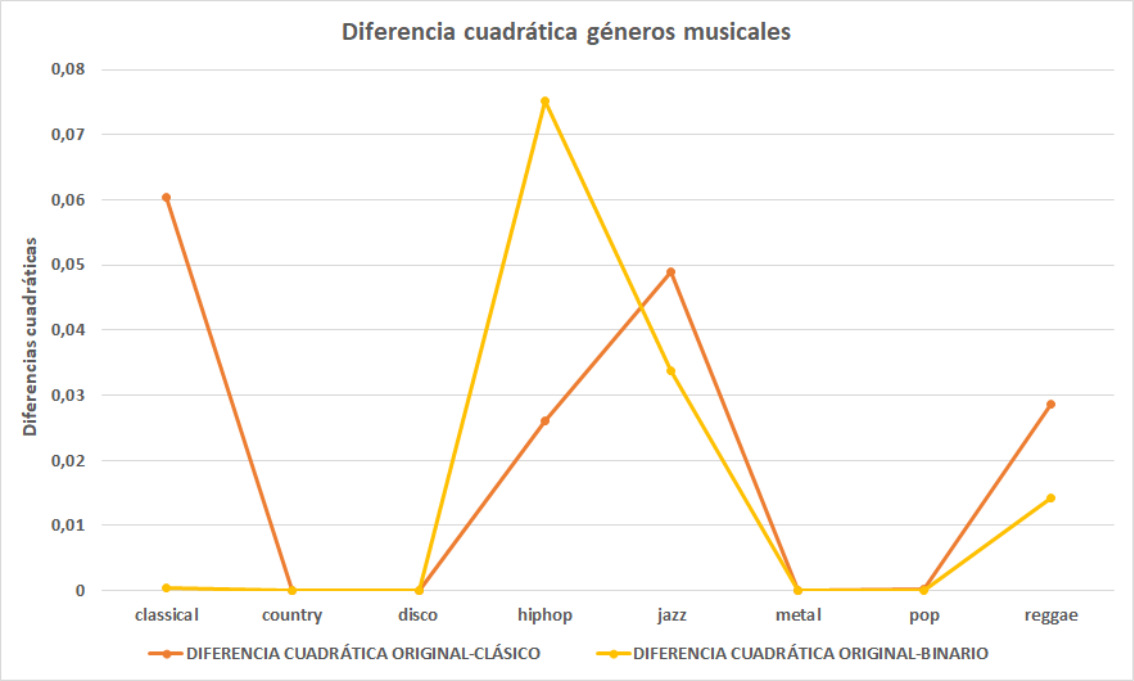


Figura 5.6: Diferencias cuadráticas en en reconocimiento de género.

Respecto a la técnica MIR utilizada, es interesante el hecho de que para el conjunto original no se haya clasificado ningún ejemplo en los géneros *country y disco*, que encontremos muy pocos ejemplos de *hiphop, metal y pop* y que los géneros *rock y blues* hayan desaparecido. Podríamos justificar esta ausencia ya que algunos de éstos no se han incluido directamente en el conjunto de datos, sin embargo, los géneros que sí se han incluido comparten muchas características con éstos, lo que podría derivar en ejemplos clasificados a partir de esos géneros. El caso que más llama la atención es el del género *pop* ya que es uno de los géneros principales del conjunto de datos. Creemos que el origen de estas limitaciones reside en las siguientes razones:

Datos de entrenamiento: El GTZAN dataset, a pesar de ser un conjunto de datos ampliamente usado cuenta con limitaciones tales como **clasificaciones erróneas** o **predicciones repetidas**. En [74] se realiza un estudio sobre éstas, así como sus utilidades, y el uso futuro que puede tener éste.

Fuentes de sonido: Como ya se mencionó en el apartado 4.3, para convertir ficheros MIDI a formato MP3 necesitamos utilizar fuentes de sonido que interpretan la información almacenada en éstos y la transformen a audio. Es posible que los instrumentos de éstas le hayan dado el carácter observado a todos los MIDI utilizados. Teniendo en cuenta que la selección es aleatoria y que cada fuente de sonidos cuenta con sus propios instrumentos que hacen que cada generación suene distinta, el problema puede residir en que dispongamos de un conjunto de fuentes de sonido muy reducido.

Realizadas las observaciones correspondientes sobre la técnica MIR aplicada, pasamos a analizar los resultados obtenidos en la encuesta musical.

5.3. Análisis encuesta musical

Antes de comenzar el análisis de los datos recogidos en la encuesta, introducimos brevemente algunos aspectos de ésta a tener en cuenta a lo largo esta sección.

El período activo de la encuesta ha sido de 3 semanas, comenzando el día 4 de agosto de 2020. Durante este periodo se han recogido 128 respuestas, de las cuales, 71 provienen de la encuesta en español y 57 de la encuesta en inglés. Según la división por grupos especificada en el apartado 4.3.3, éstos están formados por:

- **Grupo 1:** 47 respuestas.
- **Grupo 2:** 32 respuestas. 18 indican enseñanzas musicales previas y 14 autoaprendizaje.
- **Grupo 3:** 29 respuestas. 17 indican enseñanzas musicales previas y 12 autoaprendizaje.
- **Grupo 4:** 20 respuestas. 13 indican enseñanzas musicales previas y 7 autoaprendizaje.

Entre las melodías incluidas en la encuesta no existe ninguna que haya sido compuesta por un humano, habiendo obtenido todas mediante las herramientas anteriormente descritas.

Respecto a los campos opcionales, edad, estilos musicales reconocidos y opinión, realizaremos observaciones sobre los dos primeros, mientras que, las opiniones más interesantes se mostrarán en el apéndice C. Junto a estas opiniones, incluiremos algunas discusiones con los usuarios de la plataforma *reddit*.

Comenzamos el análisis mostrando los resultados por grupos, empezando por el grupo 1:

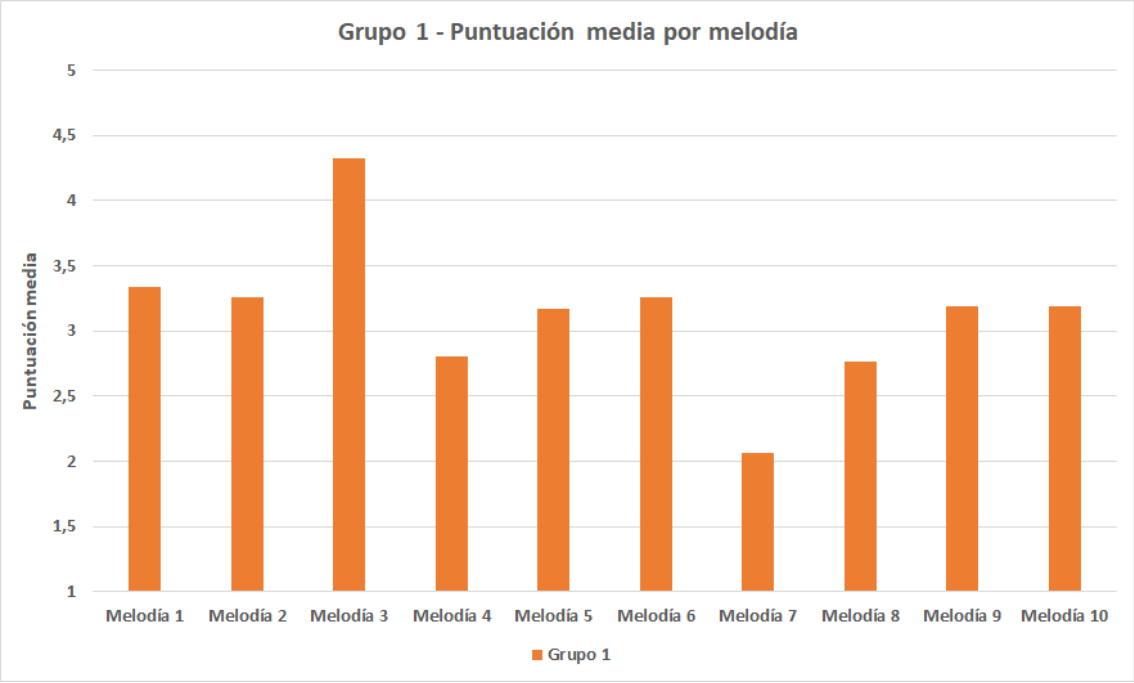


Figura 5.7: Resultados medios grupo 1 en encuesta musical.

Lo primero que observamos es que la mayoría de las melodías, exceptuando la **3** y la **7** giran en torno a la puntuación media. El hecho de que ésta sea la más frecuente nos indica una gran incertidumbre por parte los encuestados a la hora de determinar si la melodía escuchada ha sido compuesta por un humano o una máquina. Respecto a las excepciones, la melodía 3 es aquella que más se ha puntuado como generada por un ordenador, con una puntuación de '4.32' de media. La melodía 7 por su parte, obtiene la menor puntuación, con un '2.06' de puntuación media.

Los integrantes del grupo 1 en general no parecen saber distinguir con exactitud si las melodías escuchadas son humanas o no, sin embargo, sus puntuaciones están ligeramente más inclinadas hacia la composición artificial, con una media de '3.14'.

Observamos los resultados del grupo 2 en el siguiente histograma:

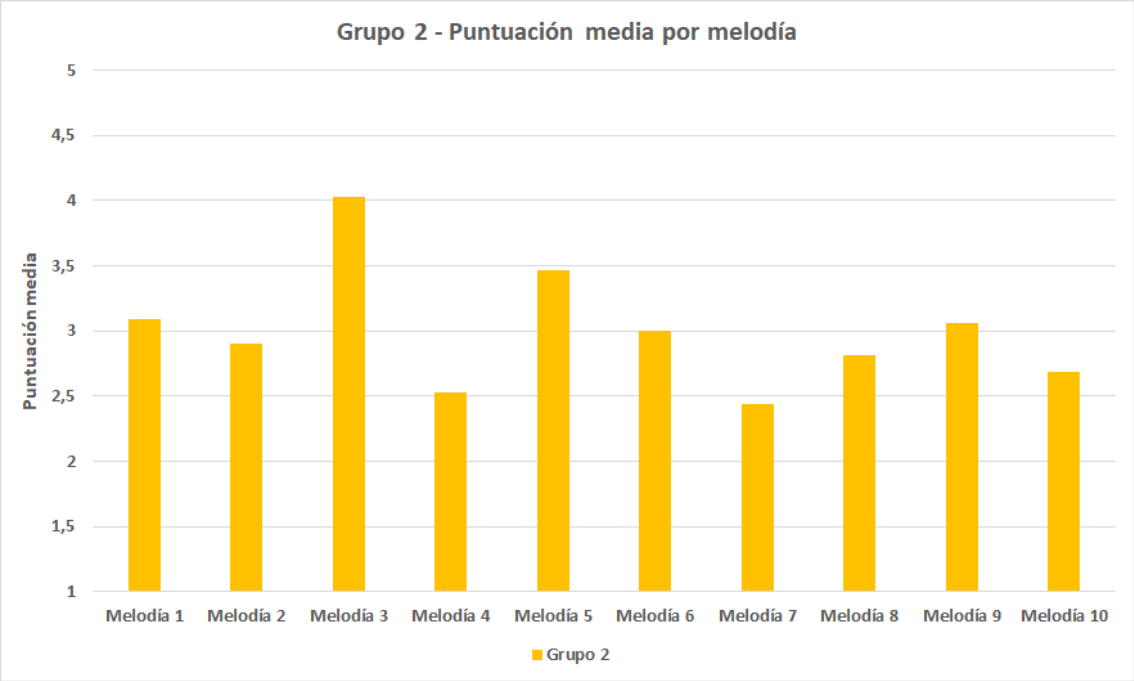


Figura 5.8: Resultados medios grupo 2 en encuesta musical.

Volvemos a ver como la mayoría de melodías giran en torno a la puntuación media, observando de nuevo la incertidumbre mencionada en el grupo anterior, sin embargo, es esta ocasión existen más melodías por debajo de la media que por encima. Entre las melodías que más se alejan de la media volvemos a observar a las melodías **3** y **7** con puntuaciones medias de '4.03' y '2.43' respectivamente. También podemos observar como las melodías **4** y **10** han sido calificadas mayormente como humanas en vez de como artificiales.

El grupo 2 muestra un comportamiento similar al del grupo 1, diferenciándose de éste principalmente en que la gran mayoría de las melodías tienen una puntuación más baja, contando con una media de '3'.

Observamos los resultados del grupo 3 en el siguiente histograma:

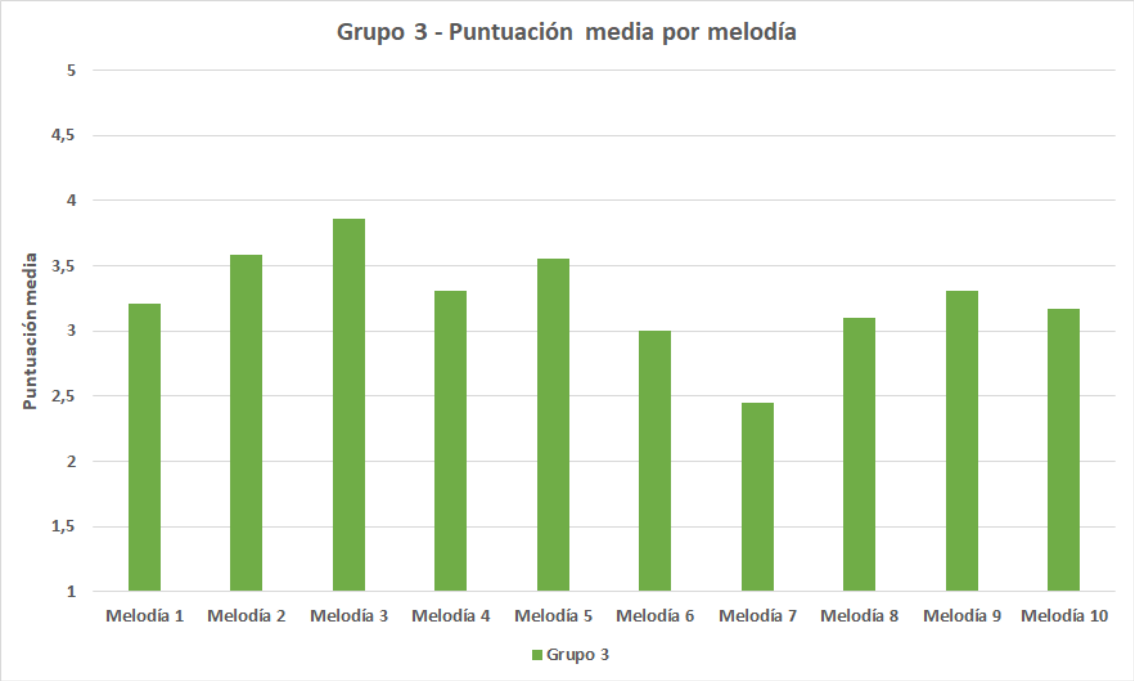


Figura 5.9: Resultados medios grupo 3 en encuesta musical.

En este caso, todas las melodías a excepción de la **7** muestran una puntuación igual o superior a la media. A pesar de que las puntuaciones obtenidas puedan implicar cierta incertidumbre, esta vez podemos ver más claramente que gran parte de los encuestados han determinado que las melodías son artificiales. La melodía **7** sigue manteniendo su status de composición humana a pesar de haber aumentado ligeramente su puntuación (’2.45’).

El grupo 3, a diferencia de los dos anteriores, obtiene unos resultados más estables que tienden a clasificar las melodías como artificiales. Interpretamos esto como una menor incertudimbre en los participantes. La puntuación media obtenida en esta ocasión es de ’3.25’

Observamos en la figura 5.10 los resultados para el grupo 4. Observamos que, por primera vez existe más de una melodía que supera la puntuación de ’3.5’, siendo éstas las melodías **1** (’4’), **2** (’3.65’) y **10** (’3.65’). En esta ocasión la melodía **3** cuenta con una puntuación mucho más baja que en el resto de grupos, con un valor de ’3.15’. Los puntuaciones medias más bajas las obtienen las melodías **4** (’2.4’) y la **7** (’2.4’). El resto de melodías giran en torno a la puntuación media por lo que parecen generar cierta incertidumbre entre los participantes.

El grupo 4 muestra variedad en sus resultados, tal y como ocurría en los grupos 1 y 2. Es el primer grupo en el que la mitad de las melodías se alejan de la puntuación media, indicando esto una mayor determinación en los encuestados a la hora de puntuar. La puntuación media de las melodías es de ’3.125’.

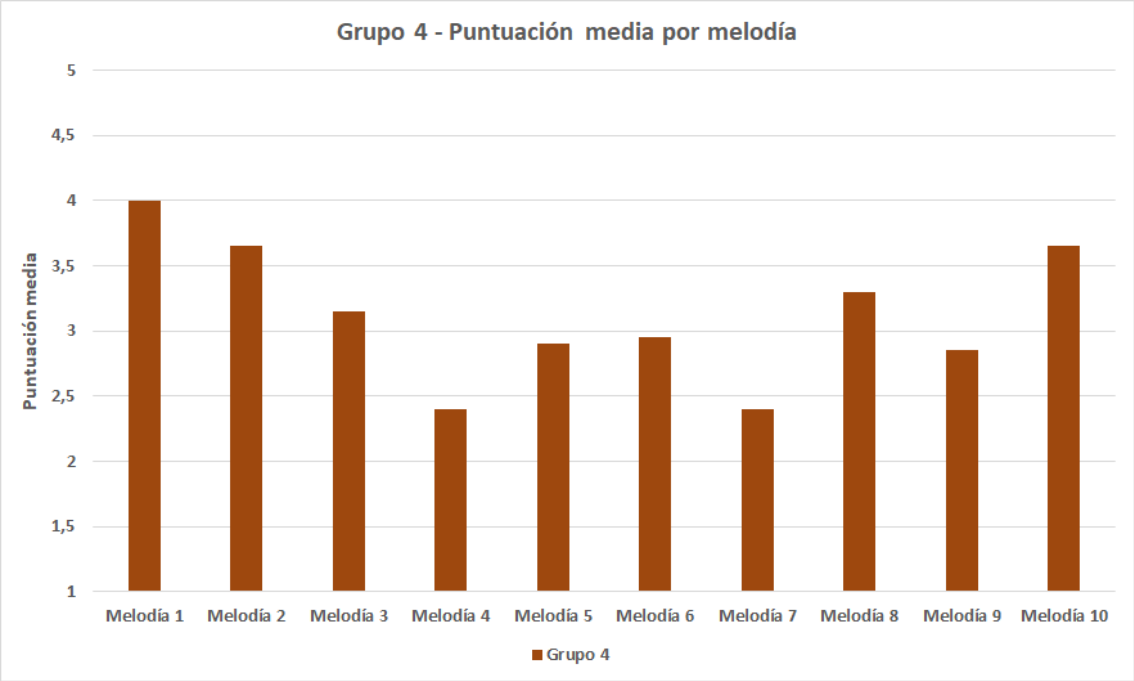


Figura 5.10: Resultados medios grupo 4 en encuesta musical.

Una vez hemos analizado individualmente cada grupo, mostramos los resultados de forma global.

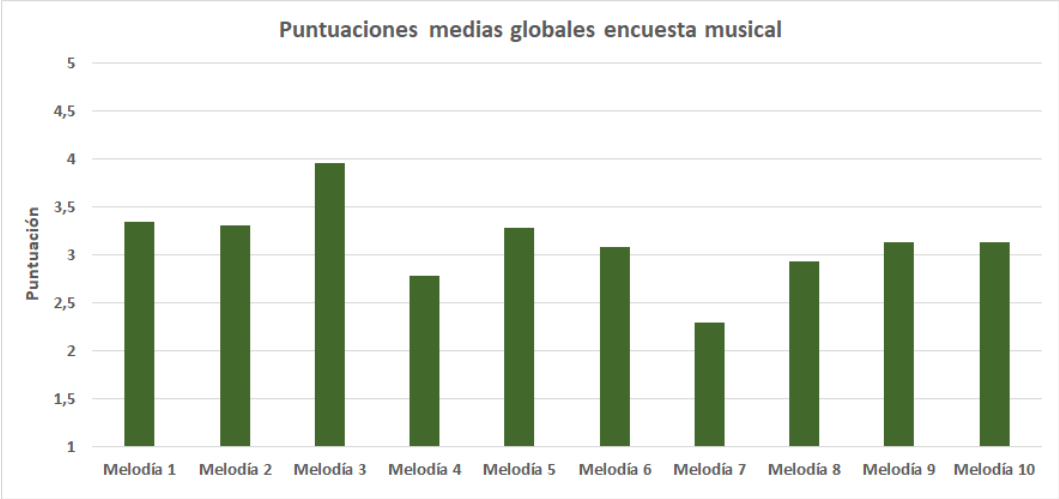


Figura 5.11: Puntuaciones medias globales en encuesta musical.

Encontramos unos resultados muy similares a los del grupo 1, con la gran mayoría de las melodías en torno al '3' de puntuación. Las melodías **3** y **7** son las únicas que se alejan de ésta, con unas puntuaciones medias de '3,96' y '2.3'. Teniendo en cuenta que el grupo 1 cuenta con más encuestados que el resto, es lógico pensar que guardará más semejanzas con éste. La puntuación media de las melodías en esta ocasión es '3.13'.

Observados los resultados globales, a modo de resumen, mostramos en la siguiente tabla todas las puntuaciones medias divididas en grupos y melodías:

Grupo\Melodía	1	2	3	4	5	6	7	8	9	10	MEDIA
Grupo 1	3,34	3,26	4,32	2,81	3,17	3,26	**2,06**	2,77	3,19	3,19	3,14
Grupo 2	3,09	2,91	4,03	2,53	3,47	3	**2,44**	2,81	3,06	2,69	3
Grupo 3	3,21	3,59	3,86	3,31	3,55	3	**2,45**	3,1	3,31	3,17	3,26
Grupo 4	4	3,65	3,15	**2,4**	2,9	2,95	**2,4**	3,3	2,85	3,65	3,13
MEDIA	3,35	3,3	3,96	2,79	3,29	3,09	**2,3**	2,94	3,13	3,13	3,13

Tabla 5.4: Puntuaciones medias en encuesta musical.

Adicionalmente, analizamos también si los datos opcionales nos pueden proporcionar información que no hayamos descubierto a partir de las puntuaciones medias. Para reflejar el comportamiento de la edad, observamos las melodías con mayor y menor puntuación media de la tabla 5.4.

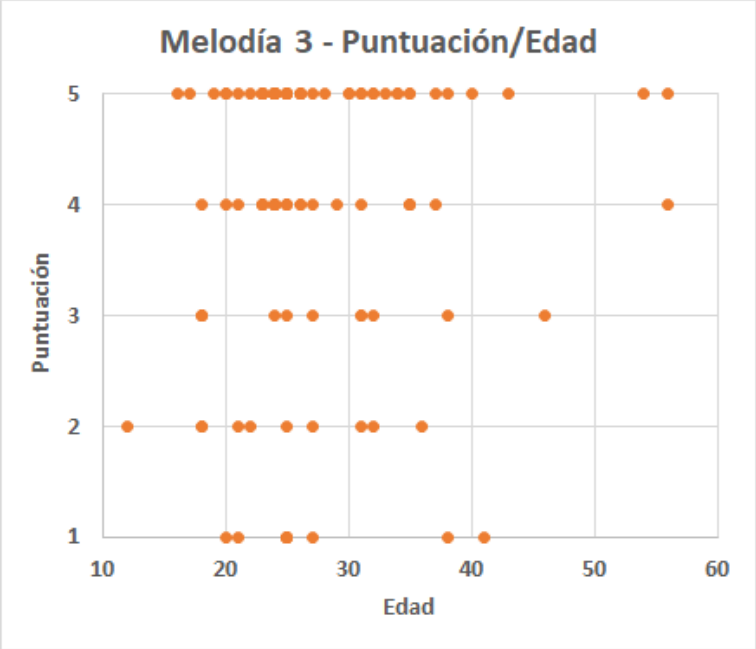


Figura 5.12: Comparativa puntuación/edad en melodía 3.

En la figura 5.12 observamos que independientemente de la edad la gran mayoría de encuestados están de acuerdo en que la melodía es artificial. Respecto a la melodía de la figura 5.13, observamos práticamente el mismo comportamiento, aunque esta vez a la inversa.

Se ha realizado esta comparación en otras melodías y no parece existir ninguna relación entre la edad y la puntuación indicada. En el caso de que existiese esta relación sería necesaria una cantidad de respuestas mayor y más variada en cuanto a edad.

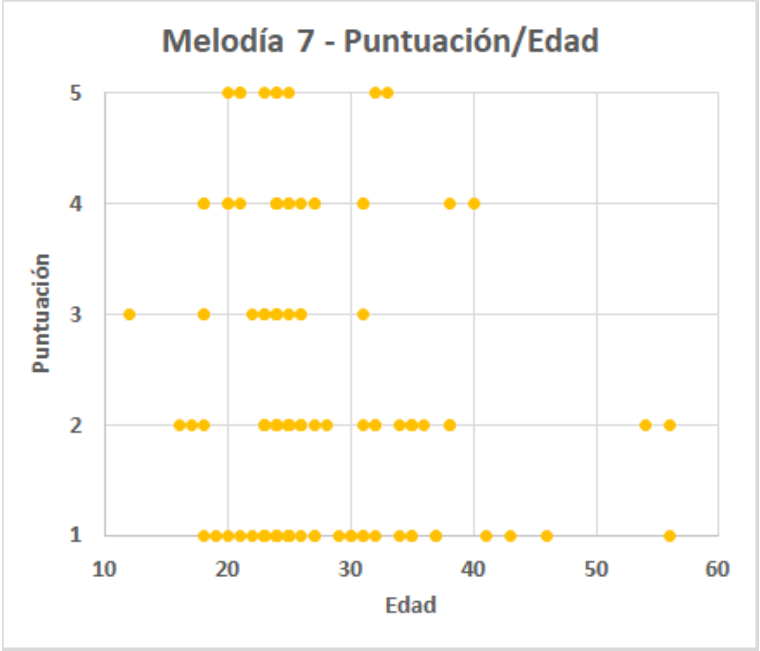


Figura 5.13: Comparativa puntuación/edad en melodía 3.

Por último, observamos los géneros que los encuestados han reconocido en las melodías a partir de la figura 5.14.

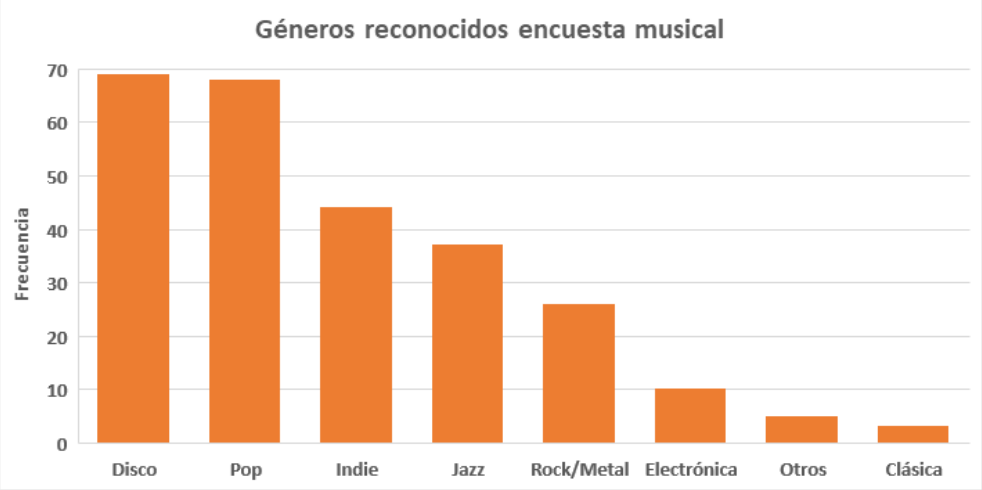


Figura 5.14: Géneros musicales reconocidos en la encuesta musical.

A pesar de que esto no nos aporte mucha información respecto a si consideran que la melodía escuchada es humana o no, nos sirve para complementar los análisis de las secciones anteriores.

Para finalizar esta sección incluimos una serie de observaciones, a modo de resumen, sobre los resultados de esta sección.

Observaciones

Las puntuaciones medias obtenidas en cada grupo resultan muy similares, encontrándose todas ellas en torno a la puntuación media. Como ya hemos ido mencionando a lo largo de la sección, esto nos hace plantearnos si realmente los usuarios han sabido llevar a cabo esta tarea, ya que, a excepción de las melodías **3** y **7**, que sí han tenido una misma clasificación por parte de la mayoría de usuarios, el resto no han sido clasificadas en ninguno de los dos tipos de composición. Podríamos justificar esto ya que el grupo 1 era el más grande y está compuesto por miembros sin conocimientos musicales previos, sin embargo, las puntuaciones medias del grupo sin conocimientos y el grupo con más experiencia son prácticamente iguales.

El factor experiencia musical, por lo tanto, parece no determinar una gran diferencia en los resultados ya que todos los grupos han presentado un comportamiento similar. Aquel que mejor ha desempeñeado esta tarea, es el grupo 3, con la puntuación media más alta de todos los grupos.

Respecto a la edad no podemos contemplarla como objeto de estudio ya que no parece existir correlación entre la puntuación y ésta. Por su parte, los géneros musicales han dado resultados completamente opuestos a los obtenidos en el análisis de reconocimiento de género, contando con los géneros *Disco* y *Pop* como los más reconocidos y el género *Clásico* como el que menos. Posiblemente la razón por la que esto ha ocurrido se deba a que las melodías presentes en la encuesta tiene instrumentos seleccionados personalmente y con una mayor variedad de estilos que aquellos presentes en los anteriores análisis. Reconocer un género musical a partir de una melodía es una tarea muy compleja, razón por la cual, las melodías de Magenta no se incluyen en el apartado 5.2. Lo que sí podemos deducir de ésto es que a partir de una melodía podemos interpretar multitud de géneros musicales simplemente cambiando el instrumento que la interpreta.

Debido a los resultados obtenidos y al formato de la encuesta, podríamos decir que esta ha resultado ser una especie de “Test de Turing” en el que los participantes no han sabido determinar si lo que estaban escuchando había sido compuesto por un humano o no.

Los análisis realizados durante este capítulo nos han permitido observar diferentes aspectos sobre los distintos tipos de generaciones obtenidas en el capítulo 4. Gracias a la información de alto nivel que nos proporcionan las técnicas MIR utilizadas y la encuesta musical llevada a cabo, el estudio realizado ha sido más sencillo de llevar a cabo y resulta más intuitivo para aquellos que se interesen por éste.

6

Discusión

El conjunto de canciones creado para el desarrollo del experimento ha resultado ser útil tanto en la fase de composición artificial como en la extracción de características. Esto ha sido posible gracias a la versatilidad que ofrece el formato MIDI. Por desgracia, debido a limitaciones técnicas el tamaño del conjunto de canciones se ha visto limitado. Hubiera sido interesante añadir más variedad a partir de la inclusión de más géneros musicales.

Las tecnologías utilizadas para la composición artificial han ofrecido unos resultados excelentes. Independientemente de los resultados obtenidos en la fase de técnicas de recuperación de información musical, estas herramientas han cumplido el propósito que se pretendía y han proporcionado generaciones con estructuras diferentes que han permitido hacer observaciones desde distintos puntos de vista.

Magenta (Google), además de los modelos que se han utilizado en este trabajo dispone de modelos enfocados a otros aspectos relacionados con la composición artificial, como la síntesis de audio (GANSynth) o la interpolación de MIDIs (MusicVAE). Esta amplia selección junto a su extensa documentación convierten a esta herramienta en uno de los mejores productos para iniciarse en el mundo de la composición artificial. El único aspecto negativo que le encontramos a Magenta es que no existen modelos capaces de generar composiciones que cuenten con múltiples instrumentos actuando de forma simultánea.

MuseGAN (Music and AI Lab, Research Center for IT Innovation, Academia Sinica) ofrece “lo contrario” a lo que podemos encontrar en Magenta. En este caso disponemos de dos modelos (que parten de la misma configuración), una documentación muy limitada y la necesidad de herramientas externas para la conversión de datos, implicando una mayor dificultad a la hora de adaptar esta herramienta a cualquier trabajo. Sin embargo, su aspecto más positivo reside en la calidad y complejidad de sus generaciones respecto a Magenta, ya que estas incorporan múltiples instrumentos que actúan simultáneamente, ofreciéndonos así, una experiencia más realista.

En cuanto a las técnicas MIR aplicadas, han cumplido el propósito que tenían, sin embargo, la herramienta de etiquetado musical ha proporcionado unos resultados más interesantes y variados que la enfocada al reconocimiento de género.

Musicnn nos ha permitido observar las "características" de nuestros distintos conjuntos de audio mediante información de alto nivel (etiquetas). Al disponer de una documentación completa y utilizar en sus modelos dos datasets populares en el ámbito del etiquetado musical, esta herramienta se convierte en una opción a tener en cuenta para futuros desarrollos.

LSTM GClassifer ha llevado a cabo la tarea de reconocimiento de género de forma incompleta. El modelo pre-entrenado del que dispone no contempla los géneros *rock y blues*, lo cual supone para nuestro trabajo una gran limitación ya que el conjunto de datos original contempla numerosos ejemplos del primero. Dada la multitud de alternativas disponibles, consideramos esta herramienta útil solamente si se utiliza con el objetivo de entrenarla sobre un conjunto de datos etiquetado. De este modo seguramente se obtuviesen resultados más completos.

Por su parte, la **encuesta musical** ha contado con una participación suficiente como para plantear un análisis sobre los resultados recogidos. Estos junto a las opiniones de los encuestados aportan distintos puntos de vista y permiten plantear discusiones sobre cómo los humanos percibimos e interpretamos la música. Esto nos permite aportar al trabajo una perspectiva completamente diferente a la obtenida mediantes las anteriores técnicas.

El análisis realizado sobre los datos extraídos mediante técnicas MIR ha demostrado que las generaciones obtenidas mantienen gran parte de la información contenida en el conjunto de datos original. La utilización del error cuadrático medio ha permitido determinar satisfactoriamente las diferencias entre los conjuntos ya que nos permitía establecer numéricamente una "distancia" sobre estos.

Respecto al **etiquetado musical**, los conjuntos procedentes de la herramienta MuseGAN son los que más se asemejan al conjunto original. El conjunto de Magenta, debido a las diferencias ya mencionadas en el apartado 5.1. Consideramos por lo tanto que, las generaciones de MuseGAN demuestran una gran semejanza respecto a los datos originales mientras que, las generaciones de Magenta, a pesar de preservar bastante información, resultan menos adecuadas para esta tarea.

En cuanto al **reconocimiento de género**, el conjunto que más información retiene es el **binario**, mientras que el **clásico** preserva una cantidad ligeramente menor. A pesar de que los ECM obtenidos no sean muy elevados, ambos conjuntos presentan diferencias remarcables respecto al conjunto original. Hay que tener en cuenta que cuatro de los ocho géneros presentados prácticamente no tiene presencia en los resultados, por lo que la distancia real entre los conjuntos es ligeramente superior. Teniendo esto en cuenta, sí es cierto que existen semejanzas y que parte de la información se mantiene, sin embargo, no podemos establecer que esta herramienta haya proporcionado unos datos tan concluyentes como los obtenidos mediante el etiquetado musical.

Los distintos factores planteados para analizar los resultados de la **encuesta musical** no han conseguido establecer diferencias sobre estos. Inicialmente considerábamos que aquellos encuestados con mayor **conocimiento musical** serían capaces de establecer más fácilmente las diferencias entre una melodía artificial y una compuesta por un humano, sin embargo, los resultados obtenidos han de-

mostrado que todos estos actúan de la igual manera frente a las melodías planteadas. Creemos que, el hecho de que las puntuaciones medias de cada grupo estén en torno a la puntuación media representa una gran incertidumbre por parte de los encuestados. La justificación que le damos a esto es que, los encuestados prefieren indicar la puntuación media antes que clasificar erróneamente las melodías. Respecto a los campos opcionales, hemos podido observar que la **edad** no parece influir en las puntuaciones proporcionadas y que, entre los **géneros reconocidos** por los encuestados se encuentran los géneros del conjunto original así como otros géneros y subgéneros que han sido clasificados como *Electrónica* a fin de simplificar los resultados.

Tal y cómo planteamos anteriormente en este trabajo, la música es un campo muy amplio en el que existen infinitas interpretaciones posibles. Cada persona forma una idea de lo que es la música a partir de su contexto musical, sus experiencias y su alrededor. El hecho de que los usuarios no hayan sido capaces de establecer diferencias entre las melodías planteadas junto a la diversidad de los resultados no hace más que demostrar que no existe una forma correcta de realizar esta tarea.

7

Conclusiones y trabajo futuro

La composición artificial sin duda supone un cambio en la música tal y como la interpretamos, sin embargo, a pesar de que las generaciones preservan una gran cantidad de información respecto al conjunto original, no podemos considerarlas mas que una herramienta en la que apoyarse a la hora de componer música. Con el desarrollo futuro de estas tecnologías será interesante volver a plantearse la misma cuestión, sin embargo, la música nunca tendrá una forma o intención definida, por lo que seguramente tendremos que esperar muchos años para ver un sistema capaz de simular música de forma realista.

Conforme al trabajo realizado, los experimentos desarrollados nos han permitido aproximarnos a la recuperación de información musical y generación musical. Han sido particularmente útiles los conjuntos de datos procedentes de **MuseGAN** ya que han resultado ser aquellos que más semejanzas presentan frente al conjunto original. **Magenta** presenta resultados excelentes, sin embargo, la comparación respecto el conjunto original no es del todo acertada por las diferencias estructurales de estos.

Los resultados obtenidos a través de la encuesta demuestran la subjetividad que supone la tarea de reconocimiento de música creada por ordenador. Lo que para unos es música para otros no, lo cual se ha podido ver en la incertidumbre que los encuestados han demostrado a la hora de determinar el origen de las melodías planteadas.

De acuerdo a los objetivos que habíamos planteado para la realización del trabajo, podemos afirmar que se han cumplido satisfactoriamente. En particular:

O-1: Se ha llevado a cabo la creación de un conjunto de canciones de géneros populares en formato MIDI que ha demostrado una gran versatibilidad adaptándose a todas las herramientas utilizadas.

O-2: Se han generado composiciones artificiales de distinta naturaleza a través de las herramientas Magenta y MuseGAN que nos han permitido plantear distintos tipos de análisis.

O-3: Se ha desarrollado una encuesta musical a través de resultados obtenidos en el **O-2** que planteaba la capacidad de los humanos de diferenciar entre una composición humana y una artificial.

O-4: A través de las técnicas MIR planteadas (etiquetado musical y reconocimiento de género) se han obtenido características de alto nivel sobre el conjunto de datos original y aquellos obtenidos mediante el **O-2**.

O-5: Mediante el análisis de los resultados obtenidos y la utilización del error cuadrático medio hemos sido capaces de establecer las diferencias y semejanzas entre el conjunto de datos original y los conjuntos obtenidos mediante generación.

7.1. Trabajo futuro

De cara a continuar con la investigación iniciada, existen multitud de aspectos a mejorar y alternativas a tener en cuenta.

En primer lugar, sería conveniente realizar una **ampliación** sobre el **conjunto de datos** que incluya más géneros musicales, equilibrio sobre el número de ejemplos de estos y un procesado de todos los MIDI contenidos a fin de eliminar ejemplos corruptos.

Añadir **más técnicas MIR** a la fase de extracción, como por ejemplo, **separación de fuentes** [75] para analizar de forma separada los elementos de cada composición, **estimación de clave** [76] o **estimación de tempo** [77], así como **distintas implementaciones de las ya implementadas**. Existen multitud de herramientas enfocadas a la recuperación de información musical que representan distintos aspectos de la música.

Plantear la **extracción de características de bajo nivel**, es decir, utilizar métricas tales como los MFCC (Mel Frequency Cepstral Coefficients), STFT (Short Time Fourier Transform) o CQT (Constant Q Transform) de forma conjunta para establecer las diferencias y semejanzas entre audio original y audio generado.

Optimizar todas las redes neuronales utilizadas a través de la configuración más adecuada para cada tarea, así como disponer de un **equipo de altas prestaciones** capaz de simplificar las tareas llevadas a cabo.

Ampliar la cantidad de fuentes de sonido en la etapa de conversión MIDI a MP3 para dotar de mayor variedad a las composiciones y así realizar una evaluación más realista de los resultados.

En resumen, la realización de este trabajo me ha demostrado que se abren numerosas alternativas de estudio, análisis y profundización en una disciplina que puede suponer un antes y un después en el sector musical. Siempre debemos tener en cuenta que estas herramientas solamente deben ser un apoyo para el compositor y nunca un reemplazo. En lo personal, las composiciones obtenidas a lo largo del trabajo me resultan de gran utilidad ya que me permiten componer con mayor facilidad en situaciones de bloqueo. Por su parte, las técnicas MIR ya se están utilizando en aplicaciones como **Spotify**, **Shazam** o **Youtube**, por lo que es cuestión de tiempo ver su aplicación en nuevos contextos.

Como ya se mencionó al comienzo del trabajo, la música está completamente influenciada por el contexto en el que se desarrolla, por lo que, es de esperar que poco a poco estas nuevas tecnologías formen parte de composiciones profesiones.

Al fin y al cabo, la evolución del ser humano y el arte siempre ha ido de la mano.

Bibliografía

[1] J. Wildridge, "Characteristics of prehistoric music: An introduction," *CMUSE*, Septiembre 2018.

[2] "Make Music and Art Using Machine Learning." `https://magenta.tensorflow.org/`. Accedido por última vez: 2020-08-02.

[3] "Mozart - Musical Game in C K. 516f*." `http://www.asahi-net.or.jp/~rb5h-ngc/e/k516f.htm`. Accedido por última vez: 2020-08-03.

[4] G. Papadopoulos and G. Wiggins, "Ai methods for algorithmic composition: A survey, a critical view and future prospects," in *AISB symposium on musical creativity*, vol. 124, pp. 110–117, Edinburgh, UK, 1999.

[5] J. D. Fernández and F. Vico, "Ai methods in algorithmic composition: A comprehensive survey," *Journal of Artificial Intelligence Research*, vol. 48, pp. 513–582, 2013.

[6] P. Worth and S. Stepney, "Growing music: musical interpretations of l-systems," in *Workshops on Applications of Evolutionary Computation*, pp. 545–550, Springer, 2005.

[7] R. Akerkar and P. Sajja, *Knowledge-based systems*. Jones & Bartlett Publishers, 2009.

[8] C. Ames, "The markov process as a compositional model: A survey and tutorial," *Leonardo*, pp. 175–187, 1989.

[9] C. Bircanoğlu and N. Arıca, "A comparison of activation functions in artificial neural networks," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2018.

[10] K. J. Hsü and A. Hsü, "Self-similarity of the"1/f noiseçalled music.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 8, pp. 3507–3509, 1991.

[11] B. Chopard and M. Droz, *Cellular automata*, vol. 1. Springer, 1998.

[12] K. Choi, G. Fazekas, K. Cho, and M. Sandler, "A tutorial on deep learning for music information retrieval," *arXiv preprint arXiv:1709.04396*, 2017.

[13] K. Choi, G. Fazekas, and M. Sandler, "Explaining deep convolutional neural networks on music classification," *arXiv preprint arXiv:1607.02444*, 2016.

[14] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2009.

[15] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling.," in *Ismir*, vol. 270, pp. 1–11, 2000.

[16] O. Douglas and O. Shaughnessy, "Speech communications: Human and machine," *IEEE press, Newyork*, pp. 367–433, 2000.

[17] G. Kour and N. Mehan, "Music genre classification using mfcc, svm and bpnn," *International Journal of Computer Applications*, vol. 112, no. 6, 2015.

[18] S. Dieleman and B. Schrauwen, "Multiscale approaches to music audio feature learning," in *Proceedings of the 14th international society for music information retrieval conference* (A. De Souza Britto Jr., F. Gouyon, and S. Dixon, eds.), pp. 116–121, Pontifícia Universidade Católica do Paraná, 2013.

[19] J. Brown, "Calculation of a constant q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, pp. 425–434, January 1991.

[20] S. Sigtia, N. Boulanger-Lewandowski, and S. Dixon, "Audio chord recognition with a hybrid recurrent neural network.," in *ISMIR*, pp. 127–133, 2015.

[21] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[22] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[23] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile networks and applications*, vol. 19, no. 2, pp. 171–209, 2014.

[24] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[25] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[26] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[27] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[28] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017.

[29] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS journal of photogrammetry and remote sensing*, vol. 145, pp. 120–147, 2018.

[30] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4438–4446, 2017.

[31] R. Hou, C. Chen, and M. Shah, "Tube convolutional neural network (t-cnn) for action detection in videos," in *Proceedings of the IEEE international conference on computer vision*, pp. 5822–5831, 2017.

[32] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in neural information processing systems*, pp. 2643–2651, 2013.

[33] K. Xu, Y. Feng, S. Huang, and D. Zhao, "Semantic relation classification via convolutional neural

networks with simple negative sampling," *arXiv preprint arXiv:1506.07650*, 2015.

[34] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[36] "Photo Editing with Generative Adversarial Networks (Part 1)." `https://developer.nvidia.com/blog/photo-editing-generative-adversarial-networks-1/`. Accedido por última vez: 2020-08-12.

[37] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1316–1324, 2018.

[38] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[39] P. Bhattacharjee and S. Das, "Temporal coherency based criteria for predicting video frames using deep multi-stage generative adversarial networks," in *Advances in Neural Information Processing Systems*, pp. 4268–4277, 2017.

[40] J. Rothstein, *MIDI: A comprehensive introduction*, vol. 7. AR Editions, Inc., 1992.

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[42] E. Waite, "Generating long-term structure in songs and stories." `https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn`, note = Accedido por última vez: 2020-08-12, 2016.

[43] F. T. Liang, M. Gotham, M. Johnson, and J. Shotton, "Automatic stylistic composition of bach chorales with deep lstm.," in *ISMIR*, pp. 449–456, 2017.

[44] V. Vakilotojar, "Magenta$_{M}odulo_{E}ncoding_{F}or_{M}idi_{P}erformances,''$ $May 2018. Accedido por ltima vez: 2020 - 08 - 12.$

[45] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics." `https://magenta.tensorflow.org/performance-rnn`, 2017. Accedido por última vez: 2020-08-12.

[46] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[47] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Demonstration of a convolutional gan based model for generating multi-track piano-rolls," *ISMIR Late Breaking/Demos*, 2017.

[48] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[49] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical distributions*. John Wiley & Sons, 2011.

[50] S. Foucart, "Hard thresholding pursuit: an algorithm for compressive sensing," *SIAM Journal on Numerical Analysis*, vol. 49, no. 6, pp. 2543–2563, 2011.

[51] H.-W. Dong and Y.-H. Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," *arXiv preprint arXiv:1804.09399*, 2018.

[52] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *19th International Society for Music Information Retrieval Conference (ISMIR2018)*, 2018.

[53] J. Pons and X. Serra, "musicnn: pre-trained convolutional neural networks for music audio tagging," in *Late-breaking/demo session in 20th International Society for Music Information Retrieval Conference (LBD-ISMIR2019)*, 2019.

[54] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging.," in *ISMIR*, pp. 387–392, 2009.

[55] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[56] R. Ruotsi, "LSTM-Music-Genre-Classification," Oct. 2019. Accedido por última vez: 2020-08-13.

[57] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[58] E. Schubert, J. Wolfe, and A. Tarnopolsky, "Spectral centroid and timbre in complex, multiple instrumental textures," in *Proceedings of the international conference on music perception and cognition, North Western University, Illinois*, pp. 112–116, sn, 2004.

[59] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai, "Music type classification by spectral contrast feature," in *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 113–116 vol.1, 2002.

[60] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," *arXiv preprint arXiv:1810.12247*, 2018.

[61] J. Shenk, "Create genre-specific melodies using TensorFlow," July 2017. Accedido por última vez: 2020-08-13.

[62] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.

[63] C. Raffel, "The Lakh MIDI Dataset v0.1," 2016. Accedido por última vez: 2020-08-13.

[64] A. Spotify, "Spotify web api," 2017. Accedido por última vez: 2020-08-14.

[65] P. Lamere, "spotipy," June 2020. Accedido por última vez: 2020-08-14.

[66] H.-W. Dong, "lakh-pianoroll-dataset," Jan. 2020. Accedido por última vez: 2020-08-14.

[67] W.-Y. Hsiao, "symbolic-musical-datasets/5-track-pianoroll," Aug. 2018. Accedido por última vez: 2020-08-14.

[68] D. Rossum and E. Joint, “The SoundFont® 2.0 File Format,” *Joint E-Mu/Creative Tech Center white paper*, 1995.

[69] D. Henningsson and F. Team, “Fluidsynth real-time and thread safety challenges,” in *Proceedings of the 9th International Linux Audio Conference, Maynooth University, Ireland*, pp. 123–128, 2011.

[70] J. Robert, “pydub,” July 2020. Accedido por última vez: 2020-08-18.

[71] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, “Pypianoroll: Open source python package for handling multitrack pianoroll,” *Proc. ISMIR. Late-breaking paper;[Online] https://github. com/salu133445/pypianoroll*, 2018.

[72] D. Bryant, “MIDI to Audio Conversion With Python,” Aug. 2013. Accedido por última vez: 2020-08-18.

[73] A. Joshi, S. Kale, S. Chandel, and D. K. Pal, “Likert scale: Explored and explained,” *Current Journal of Applied Science and Technology*, pp. 396–403, 2015.

[74] B. L. Sturm, “The state of the art ten years after a state of the art: Future research in music information retrieval,” *Journal of New Music Research*, vol. 43, p. 147–172, Apr 2014.

[75] W. Lim and T. Lee, “Harmonic and percussive source separation using a convolutional auto encoder,” in *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1804–1808, IEEE, 2017.

[76] F. Korzeniowski and G. Widmer, “End-to-end musical key estimation using a convolutional neural network,” in *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 966–970, IEEE, 2017.

[77] S. Böck, F. Krebs, and G. Widmer, “Accurate tempo estimation based on recurrent neural networks and resonating comb filters.,” in *ISMIR*, pp. 625–631, 2015.

Apéndices

A

Script python para extracción de géneros

Código A.1: En esta figura se presenta el código correspondiente al script getGenres.py (1)

```
1  import sys, os, shutil, json
2  import spotipy as sp
3  import spotipy.util as util
4
5  CLIENT_ID = # Obtener mediante cuenta Spotify API
6  CLIENT_ID_SECRET = # Obtener mediante cuenta Spotify API
7  RED_URI = # Obtener mediante cuenta Spotify API
8  MIDI_IN_PATH = # Directorio donde se encuentran los midis sin separar
9  MIDI_OUT_PATH = # Directorio donde se separarán los midis
10
11 def loginSpotify():
12     return util.prompt_for_user_token('Nosez', 'user-read-private',
13                                       client_id=CLIENT_ID,
14                                       client_secret=CLIENT_ID_SECRET,
15                                       redirect_uri=RED_URI)
16
17 def getGenresFromDir(directory, authToken):
18     artists = [item for item in os.listdir(directory) if not item.startswith('.')]
19     session = sp.Spotify(auth=authToken)
20     genres = {}
21
22     for i, artist in enumerate(artists):
23         try:
24             results = session.search(q=artist, type='artist', limit=1)
25             items = results['artists']['items']
26             genre_list = items[0]['genres'] if len(items) else items['genres']
27             genres[artist] = genre_list
28         except Exception as e:
29             print("INFO:␣", artist, "not␣included:␣", e)
30
31     genres_flattened = [item for sublist in list(genres.values()) for item in sublist]
32
33     return genres
```

Código A.2: En esta figura se presenta el código correspondiente al script getGenres.py (2)

```
35  def getArtistsByGenre(genre, spotifyGenres):
36      return [artist for artist, gs in spotifyGenres.items() if genre in str(gs)]
37
38  def createGenreDir(genres, spotiGenres, directory):
39      genre_data = {}
40
41      if not genres:
42          return
43
44      for gen in genres:
45          genre_data[gen] = getArtistsByGenre(gen, spotiGenres)
46
47      # Copy artists to a genre-specific folder
48      for genre, artists in genre_data.items():
49          try:
50              for artist in artists:
51                  _genre = genre.replace('␣','_').replace('&','n')
52                  shutil.copytree(os.path.join(directory, artist), os.path.join(os.getcwd(), 'genres', _genre,
                        artist))
53          except Exception as e:
54              print(e)
55
56  if __name__ == '__main__':
57      genres = [] # Géneros a separar
58
59      # Lectura y escritura de géneros del directorio pasado por argumento
60      if len(sys.argv) == 2:
61          dir = sys.argv[1]
62
63      # Lectura de géneros desde fichero y separación de midis por género/s
64      elif len(sys.argv) > 3:
65          dir = sys.argv[1]
66          file = sys.argv[2]
67          genres = sys.argv[3:]
68
69      # Lectura y escritura de géneros del directorio 'clean_midi' (Ejemplo base)
70      else:
71          dir = MIDI_IN_PATH
72
73      if file is not None:
74          with open(file) as jsonData:
75              spotiGenres = json.load(jsonData)
76
77      else:
78          authToken = loginSpotify()
79          spotiGenres = getGenresFromDir(dir, authToken)
80
81          json = json.dumps(spotiGenres)
82          f = open("genres.json", "w")
83          f.write(json)
84          f.close()
85
86      createGenreDir(genres, spotiGenres, dir)
```

B

Gráficas MIR completas

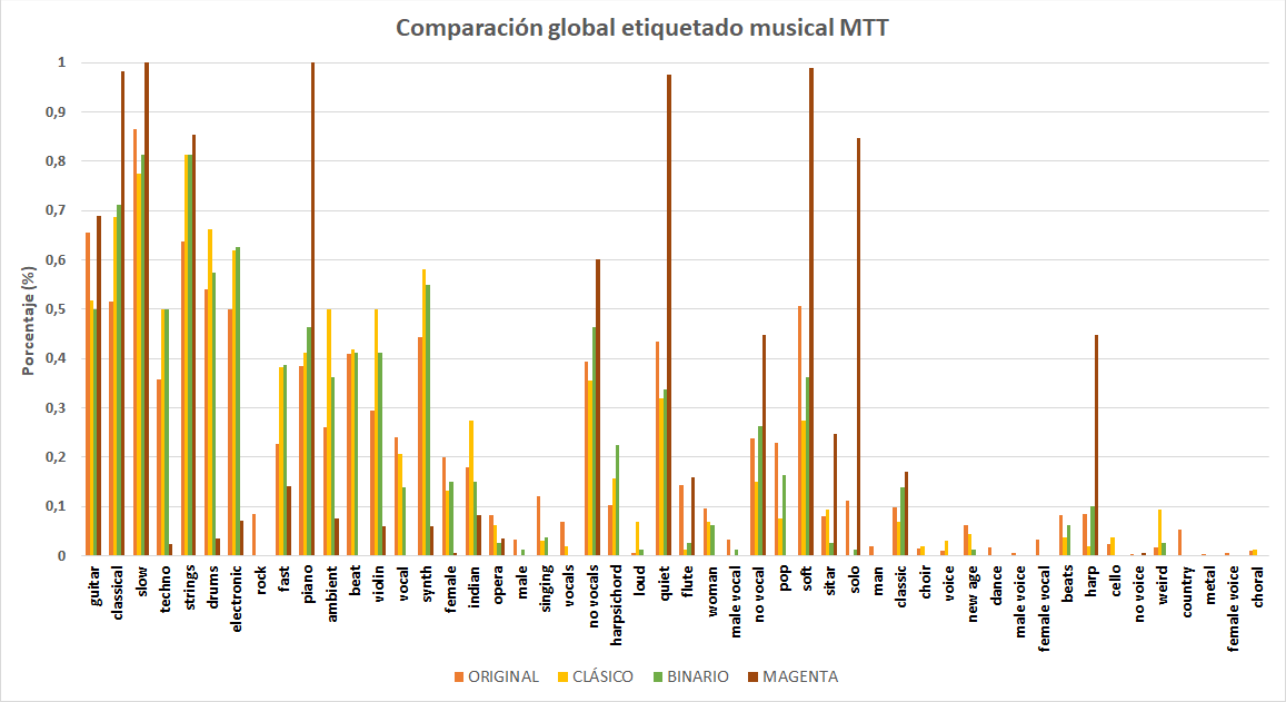


Figura B.1: Comparación global completa en etiquetado musical MTT.

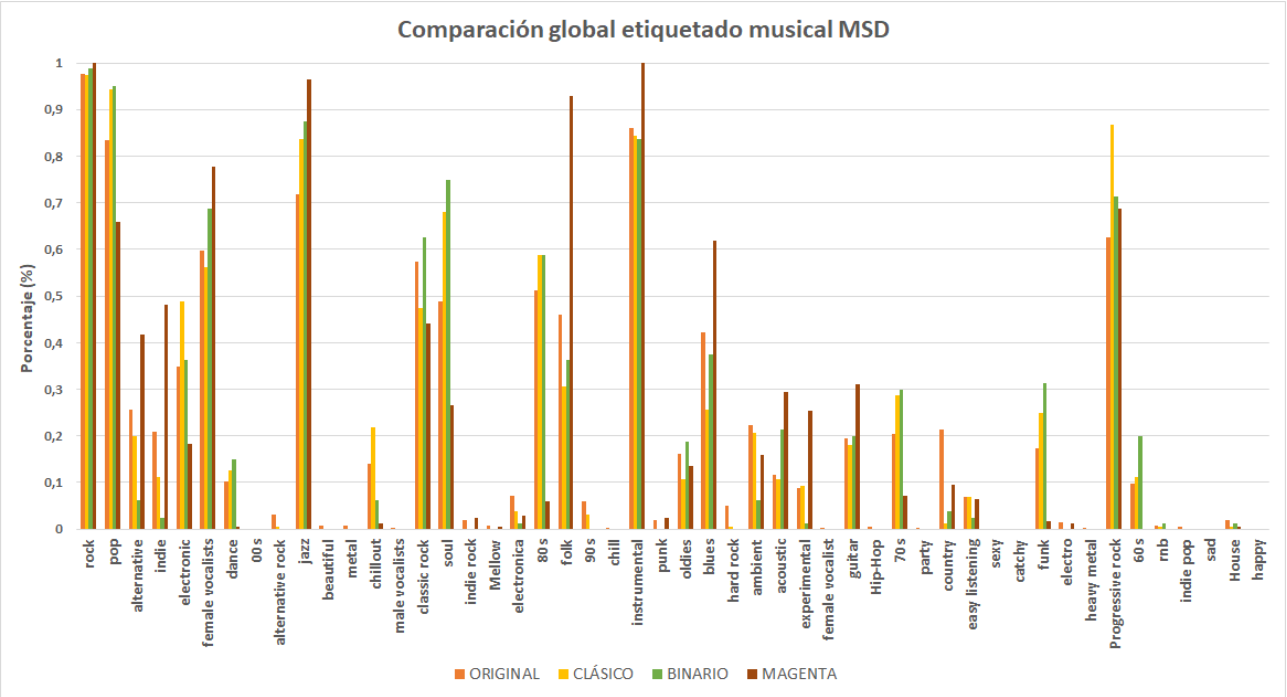


Figura B.2: Comparación global completa en etiquetado musical MSD.

C

Opiniones y discusiones sobre encuesta musical

Comentarios procedentes de la encuesta en español:

"Quizá esto sea un error mío, pero el criterio que he seguido a la hora de valorar si era una composición humana o de una máquina era el hecho de ver si mostraba y se mantenía una misma armonía en toda la pieza. Aquellas con notas disonantes he tendido a ponerlas como procedentes de máquinas, mientras que las que se mantenían congruentes durante todo el tiempo las he identificado como humanas."

"La verdad es que las composiciones son tan simples que una máquina las habría podido escribir todas ellas. Es casi imposible hacer un criterio"

Comentarios procedentes de la encuesta en inglés:

"The synth used in some melodies heavily affected what musical styles fitted the most."

"Perhaps it might be better if these were played with one or two instruments, it might be easier to analyze if it sounds human or machine based on musical composition."

"As a person with zero knowledge of music composition I was deciding if something is human or not based on how weird the music sounded. If it didn't sound weird I thought human, weird it must be AI. If it sounded too weird though I thought it was a human, like the music that was cutting off the notes."

"Done! I wonder though, wouldn't the formulation of the survey introduce a bias ? It may totally just be personal but the more artificial compositions I listened to, the more inclined I was to say the next one was a human composition. The plot twist kinda felt like a tricky teacher giving out a MCQ where all answers are A ;"

"I judge the composition either being human composed or computer composed by the range and the dynamic of the sound. If the range and dynamic is fairly tight, I'd say it's computer generated sequence."

"I don't think you can conclude anything from these samples. Most are just monophonic quantized constant velocity lines which could be easily generated by a rather simple algorithm; no human could play these lines this way, even if it was a human composer. An algorithmic composition and performance

should strive to sound like a human performance, with dynamism and feeling. Your samples seem to go the other way: even the human composition/performances (if any) sound sterile and robotic. Allow me to point out that I have been a practicing jazz musician for forty years."

"For what it's worth, I think the pieces that are the most convincingly 'human' sounding to me generally include some sort of repetition motif, that helps 'ground' the piece without letting it get to weird sounding - Test (5), Test (7), Test (3), Test (9) - it sort of unifies the whole thing. Also, ones that progress outwards from a starting point, then comfortably circle around to a resolution. Also the ones that are too noisy (like Test (10)) just sound like no human would ever choose to make that sound, so."


UAM
UNIVERSIDAD AUTONOMA
DE MADRID